VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

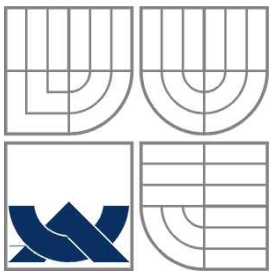
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

LOGICKÁ HRA KAKURO

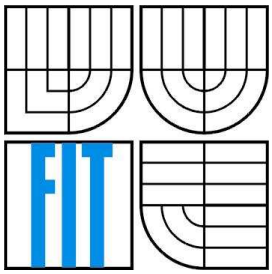
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JÁN HODÁŇ
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

LOGICKÁ HRA KAKURO

KAKURO PUZZLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JÁN HODÁŇ
AUTHOR

VEDOUCÍ PRÁCE — ING. EVA ZÁMEČNÍKOVÁ
SUPERVISOR

BRNO 2010

Zadání bakalářské práce/10049/2009/xhodan05

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2009/2010

Zadání bakalářské práce

Řešitel: **Hodáň Ján**
Obor: Informační technologie
Téma: **Logická hra Kakuro**
Kakuro Puzzle
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s pravidly hry Kakuro a jejím řešením.
2. Nastudujte potřebné kapitoly z tvorby uživatelského rozhraní.
3. Navrhněte algoritmus pro generování hracích polí s různou úrovní obtížnosti řešení a implementujte jej.
4. Dle pokynů vedoucího vytvořte aplikaci s grafickým uživatelským rozhraním, která uživateli umožní řešit hru dané složitosti.
5. Vytvořte aplikaci, která vyřeší dané zadání.
6. Zhodnoťte a diskutujte dosažené výsledky.

Literatura:

- Kakuro.com, the home of Kakuro (cross sums) on the internet [online] [cit. 1.11.2009]. Dostupné na: <http://www.kakuro.com/>
- Michael Mepham, Velká kniha Japonských rébusů, Nakladatelství BB/Art, 2007, ISBN: 978-80-7381-100-6
- Kakuro - Wikipedie, otevřená encyklopedie [online][cit. 1.11.2009]. Dostupné na <http://cs.wikipedia.org/wiki/Kakuro>

Při obhajobě semestrální části projektu je požadováno:

- Body 1., 2. a část bodu 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zámečníková Eva, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2009
Datum odevzdání: 19. května 2010

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce stručně popisuje historii logiky a logických her. Podrobně vysvětluje pravidla a způsoby řešení hry kakuro. Do hloubky bude rozebrána problematika a způsoby implementace této hry.
V posledních kapitolách bude vysvětlen způsob, kterým je logická hra implementována a způsoby testování.

Abstract

This thesis deals with history of logic and logic games. In details deals with rules and options of solving logic game Kakuro. It analyze the problem of implementation of this game to the computer algorithm. In the last chapter is explained how is logic game implemented and tested.

Klíčová slova

logika, hra, kakuro, křížovka, generátor, řešitel, program, algoritmus

Keywords

logic, game, kakuro, puzzle, generator, solver, program, algorithm

Citace

Hodáň Ján: Logická hra Kakuro, bakalářská práce, Brno, FIT VUT v Brně, 2010

Logická Hra Kakuro

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Evy Zámečníkovej.
Ďalšie informácie som čerpal z internetových stránok a odbornej literatúry. Všetku použitú literatúru a ďalšie podkladové materiály, ktoré som použil uvádzam v súpise bibliografických citácií.

……
Ján Hodáň
Dátum 20.4.2010

Poďakovanie

Týmto by som veľmi rád poďakoval Ing. Eve Zámečníkovej za jej čas, trpezlivosť, ochotu a cenné pripomienky, ktoré mi pomohli pri vypracovaní tejto bakalárskej práce. Taktiež by som rád vyjadril svoje poďakovanie svojej rodine a blízkym za stálu podporu.

Obsah

1 Úvod

Ako názov napovedá hlavným obsahom tejto práce bude logická hra kakuro. Okrem iného táto práca stručne popisuje históriu logiky a logických hier. Podrobne budú vysvetlené pravidlá a spôsoby riešenia hry kakuro. Do hĺbky rozoberám problematiku a spôsoby implementácie tejto hry. Dokument je rozdelený do niekoľkých kapitol, ktoré majú čitateľovi pomôcť pochopiť celkovú podstatu logických hier a čo je podstatnejšie objasniť problematiku implementácie logickej hry kakuro. Predtým, než začneme popisovať samotnú hru, rozdelíme celý dokument do jednotlivých kapitol.

- Prvá kapitola obsahuje neformálny úvod do problematiky a zoznámenie sa s obsahom práce.
- Úlohou druhej kapitoly je objasniť čitateľovi históriu logiky a logických hier.
- V tretej kapitole uvediem zopár najznámejších logických hier, ktoré existujú. Popíšem pravidlá a zameriam sa na štýl a prevedenie hry do softvérovej podoby.
- Od štvrtej kapitoly sa začnem venovať výlučne hre kakuro. Popíšem pravidlá hry a ukážem jednoduché metódy riešenia.
- Piata najrozsiahlejšia kapitola bude venovaná návrhu algoritmu. Podrobne rozoberiem metódy implementácie algoritmu, uvediem jednotlivé výhody a nevýhody jednotlivých spôsobov.
- V šiestej kapitole popíšem spôsob implementácie. Odôvodním, ktoré metódy som použil a prečo som ich použil.
- Siedma kapitola popisuje spôsoby a metódy testovania aplikácie. Budú rozobrané spôsoby optimalizácie užívateľského rozhrania.
- V ôsmej kapitole zhodnotím postup a výsledky riešenia.

2 História logiky a logických hier

2.1 Logika, hry a ich história

Spojenie medzi logikou a hrami siaha hlboko do minulosti. Ak uvažujeme, že hrou je debata, potom prvým priekopníkom logických hier je samotný Aristoteles. Práve on sa ako prvý pokúšal spojiť logiku a debatu do akejsi hry, v ktorej sú určité pravidlá. Pri dodržaní správneho postupu ním navrhnutého môžete vyhrať. Všetko je popísané v jeho spisoch o sylogisme. Toto dielo ostalo dlho zabudnuté až do polovice dvadsiateho storočia, kedy Charles Hamblin znovu objavil spojenie medzi dialógom a "pravidlami rozumného odvodzovania". Na tento objav nadviazal Paul Lorenz, ktorý aj na základe Hamblinovych výskumov postavil základy logiky(na základe štúdii z [4]).

Matematická teória hier bola nájdená začiatkom 20 storočia. Samotné logické hry, ako ich poznáme teraz, sa začali objavovať až v druhej polovici dvadsiateho storočia. Za ich zakladateľov, pre ich prínos do logiky, sa považujú ľudia, ktorí pôsobili už 50 rokov predtým, ako napríklad: J. Kemény, J. C. C. McKinsey, J. Neumann, W. Quine, J. Robinson a mnohí ďalší. Napokon v roku 1953 David Gale a Frank Stewart spojili sily a výsledkom ich snaženia bola práca, ktorá pojednáva o teórii množín a hier(na základe štúdii z [4]).

Logické hry to spočiatku vôbec nemali ľahké. Do roku 1950 boli zavrhované. Mnohí matematici považovali logické hry len ako povrchný prístup k logike. Avšak tento názor sa postupom času strácal a v druhej polovici dvadsiateho storočia sa základom logiky stáva technika, ktorá sa čoraz viac a viac objavuje a používa v odborných logických dokumentoch (na základe štúdii z [4]).

2.2 Logické hry

S názvom logické hry sa stretávame až v polovici dvadsiateho storočia. S prvým predchodcom súčasných číslicových krížoviek sa môžeme stretnúť už koncom devätnásteho storočia. V tomto období tvorcovia krížoviek začínajú experimentovať s upravovaním "magických štvorcov" (vysvetlím nižšie) odobraním niektorých čísel. Istý francúzsky denník so sídlom v Paríži, " *La France"* 6. júla v roku 1895 ako prvý uverejnil číslicovú krížovku - magický štvorec o veľkosti 9x9 so štyrmi podštvorcami o veľkosti 3x3. Tento štvorec bol čiastočne vyplnený a existovali pravidlá pre doplňovanie ostatných čísel, pričom bolo zaručené jediné riešenie. (Obr. 2.1)

DIVERTISSEMENTS QUOTIDIENS

N° 3879 — CARRÉ MAGIQUE DIABOLIQUE
Par M. B. Meyniel

Compléter le carré ci-dessous en employant les neuf premiers nombres chacun neuf fois de manière que les horizontales, les verticales et les deux grandes diagonales donnent toujours à l'addition le même total.

7	8	9	1	2	3	4	5	6
3				4				8
5				9				1
8				3				4
1	2	3	4	5	6	7	8	9
6				7				2
9				1				5
2				6				7
4	5	6	7	8	9	1	2	3

Ce carré devra être diabolique, c'est-à-dire que le carré restera magique si l'on place une ligne horizontale ou une colonne verticale à la suite de toutes les autres.

N° 3865 — MATHÉMATIQUES
Par M. Adolphe R.

Solution

Le marchand a vendu 27,075 vases ; le 95e jour, le dernier, il a vendu 567 vases.

Solutions justes

MM. Améthyste ; un chercheur ; Paul et Jules Duplant ; Albert Labarre ; L. Grobet ; C. Gerbaulet.

Les solutions et les envois de problèmes inédits doivent être adressés, dans la huitaine, au rédacteur soussigné.

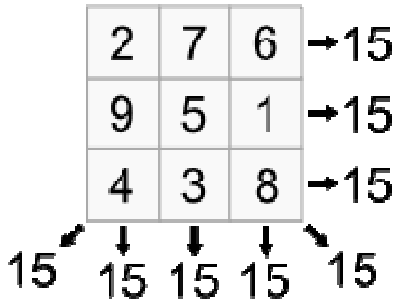
FÉLIX ANDRÉ.

Obr. 2.1.: Prvá numerická krížovka uverejnená vo Francúzskom denníku *La France* z[5])

Napriek podobnosti so súčasnými logickými hrami na vyriešenie prvej "logickej hry" boli viac ako logické schopnosti potrebné znalosti z aritmetiky.
Táto doplňovačka vychádzala každý týždeň a bola súčasťou francúzskych novín zhruba desať rokov, no počas prvej svetovej vojny sa vytratila (na základe štúdii z [5]).

2.3 Vývoj logických hier (Magický štvorec)

Predchodcom a praotcom číselných logických hier ako je kakuro, futoshiki či sudoku sa považuje práve magický štvorec. Magický štvorec (Obr. 2.3) je vlastne štvorcové pole, v ktorom platia nasledujúce pravidlá:
- vo štvorci stupňa n sa bude nachádzať n*n čísel ($n \geq 1$, n != 2)
- vo štvorci sa nesmú vyskytovať dve rovnaké čísla
- súčet v každom riadku musí byť rovnaký
- súčet v každom stĺpci musí byť rovnaký
- súčet na obidvoch diagonálach musí byť rovnaký (výpočet súčtu M=n(n*n+1)/2)



Obr. 2.3.: Magický štvorec (prevzaté z [13])

Magický štvorec sa dostaval do povedomia matematikov, astronómov, filozofov, šamanov a sektárov postupne po celej zemi (Perzii, Číne, Egypte, Arábii, Indii a nakoniec i v Európe).
Každý v ňom dokázal nájsť skryté symboly a výpočty či už pravdivé alebo vymyslené. Dôkazy o tom sa našli v knihách, maľbách a sochách po celej zemi (na základe štúdií z [13]).

Obr. 2.2 Magické štvorce rôznych kultúr (prevzaté z [13])

3 Obdoby logických hier

V súčasnej dobe sú logické hry veľmi obľúbeným a rýchlo rozvíjajúcim sa fenoménom. Od prvého uverejnenia v 1895 prvej číselnej krížovky po súčasnosť sa toho veľa zmenilo. Dnes už existujú stovky rôznych číselných logických hier. Par najznámejších si teraz ukážeme a uvediem aj stručný popis riešenia. Zamerame sa taktiež na štýl a prevedenie hry do softvérovej podoby. Z programátorského hľadiska je potrebné testovaním zistiť, aké spôsoby implementácie danej hry existujú. Ďalej je potrebné dbať na to, aby daná hra nebola len ďalšou obdobou už existujúcej varianty a mala v sebe niečo nové a výnimočné.

3.1 Sudoku

Pôvodom japonská hra sudoku je v súčasnej dobe najznámejšou logickou hrou vôbec. Hra bola popularizovaná už v roku 1986 japonskou firmou Nikoli. V roku 2005 sa stala medzinárodným hitom. Azda neexistuje človek, ktorý by ju aspoň raz v živote nehral. [5]

Klasické prevedenie hry má tvar štvorca o rozmeroch 9x9 políčok. Toto hracie pole je rozdelené na podštvorce o rozmeroch 3x3 políčok. Hovorím o klasickom prevedení hry, keďže existuje mnoho tvarov a variantov tejto hry. Napriek tomu, že tvar a prevedenie hry sa mení, hlavná myšlienka, teda pravidlá zostávajú rovnaké.

		4		7			3	
6					8	1		7
	1	3						5
	3	8	7	1				
				5	9	3	2	
5						2	1	
1		7	8					6
	9			6		7		

Obr.3.1.: Príklad zadania hry sudoku.

Japonský názov „soo-doh-ku“ napovedá (sudoku, v preklade „ jedno číslo“). Pre nájdenie riešenia zadania, musíme doplniť čísla z množiny {1,2,3,4,5,6,7,8,9} tak, aby sa v každom riadku, stĺpci a podštvorci nachádzalo práve jedno číslo. Aj vďaka takto jednoduchým pravidlám má táto hra taký veľký úspech (na základe štúdií z [5]).

Softwarové implementácie, ktoré môžeme nájsť na internete:

Obr.3.2.: 3D sudoku

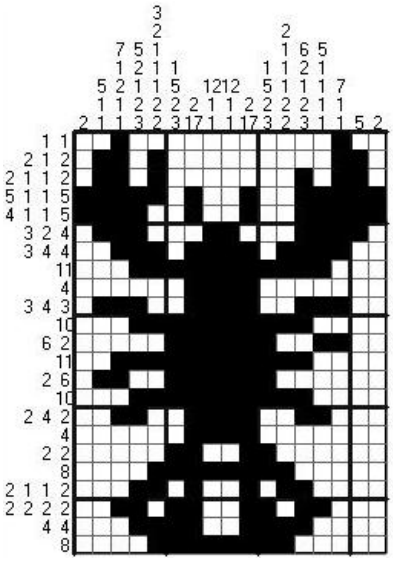
Obr.3.3.: SudokuQuest

3.2 Maľovaná krížovka

Maľovaná krížovka taktiež známa ako Nonogram alebo Griddlers je taktiež ako hra sudoku pôvodom japonská hra. Za zakladateľa, “vynálezcu“ hry sa považuje Tetsuya Nishio. Tento japonský profesionálny krížovkár v roku 1987 ako prvý uverejnil v časopisoch hru pod názvom „Paint by Numbers“. Ako každá logická hra aj táto prešla určitým vývojom. Do klasickej podoby, ako ju

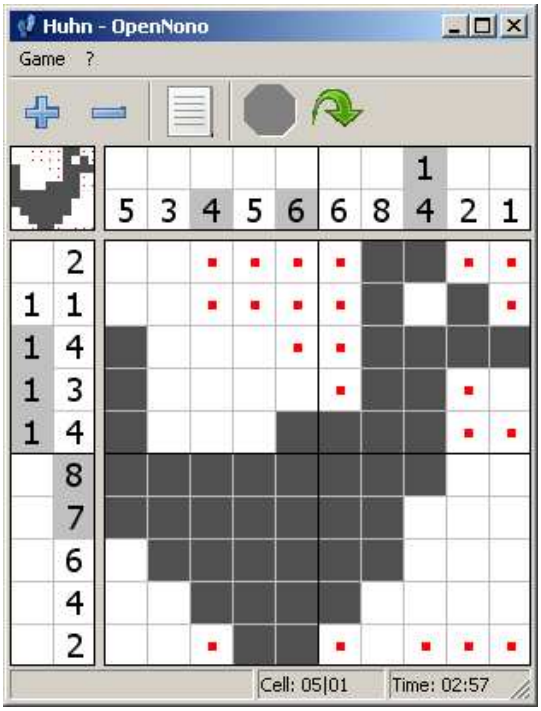
poznáme teraz, sa dostala v roku 1993 a aj u tejto hry vznikli rozličné varianty a prevedenia. (na základe štúdií z [15]).

Tvar krížovky tvorí štvorcová sieť. Nad ňou a na ľavej strane sa nachádzajú čísla, ktoré nám hovoria, akým spôsobom a kde máme vymaľovávať štvorce na štvorcovej sieti. Všimnime si prvý riadok. Na ľavej strane sú dve číslice 1,1. To znamená , že v tomto riadku vyfarbíme dve skupiny štvorcov o dĺžke jedna 1. Všimnime si prvý stĺpec. Je tam číslo 2, to znamená, že v tomto stĺpci sa bude vyskytovať jediná skupina štvorcov o dĺžke 2.

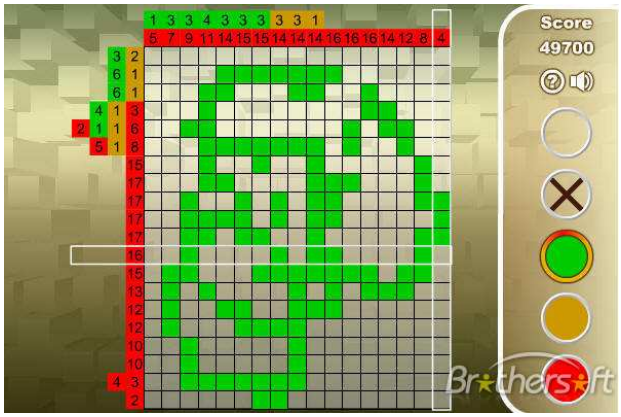


Obr. 3.4.: Maľovaná krížovka

Takýmto spôsobom sa dopracujeme k riešeniu hry. Na rozdiel od iných hier nás okrem dobrého pocitu, že sme zvládli vyriešiť logické zadanie, poteší aj výsledný obrázok.



Obr. 3.5.: OpenNono

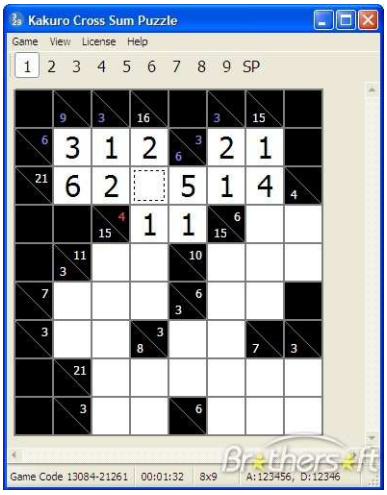


Obr. 3.6.: Nonogram 1.1.0

3.3 Kakuro

Opisu a pravidlám hry kakuro sa budeme podrobne zaoberať v nasledujúcej kapitole 4. V tejto kapitole si v krátkosti ukážeme zopár softwarových aplikácií hry Kakuro. Zameriame sa na výhody a nevýhody rôznych grafických rozhraní, ktoré nám pomôžu pri vytváraní vlastnej varianty algoritmu tejto hry.

Prvou testovanou aplikáciou bola hra Kakuro Cross Sum Puzzle. Hneď pri spustení si každý užívateľ môže všimnúť jednoduchý štýl aplikácie. Okrem toho, že daná aplikácia obsahuje čiastočnú kontrolu správnosti vyplnenia polí (kontroluje sa duplikácia čísla v riadku a stĺpci a súčet vyplnených čísel) hneď za chodu, je užívateľ odkázaný sám na seba. Nemá žiadne ďalšie možnosti pomocníka či už to zobrazenie kombinácie súčtov alebo skutočné overenie vložených číslic.



Obr. 3.7.: Kakuro Sum Puzzle

Obr. 3. 8.: Printscreen hry Kakuro Master

Ďalšou testovanou aplikáciou je hra s názvom Kakuro Master. Až na farebnosť sa na prvý pohľad veľmi neodlišuje od prvej hry. No zdanie klame. Užívateľ si už po krátkom hraní všimne skvelú vlastnosť pomocníka rýchlo a prehliadne zobrazovať kombinácie súčtov(vysvetlene v kapitole 5.1). Táto vlastnosť ako aj jednoduché ovládanie zaručujú výborný pocit z hry. Nevýhodou je trocha málo možností, či už pomocníka nápovedi, alebo absencia hrania hry výlučne myšou.

Poslednou testovanou aplikáciou je hra s názvom Kakuro dream. Dominantnú úlohu v tejto hre plní príjemný vzhľad aplikácie a ukľudňujúco pôsobiace pozadie. Ovládanie hry je vytvorené veľmi intuitívne a jednoducho, taktiež možnosti pomocníka sú rozsiahle. Dá sa povedať, že táto hra až na zopár rušivých prvkov ako je napríklad vyskakovanie okna pri nápovede číselných kombinácií nemá žiadne výrazné nedostatky. Splňa väčšinu požiadaviek na „User friendly“ (priateľské prostredie) a dbá na pohodlie užívateľa.

Obr. 3. 9.: Print screen hry Kakuro dream

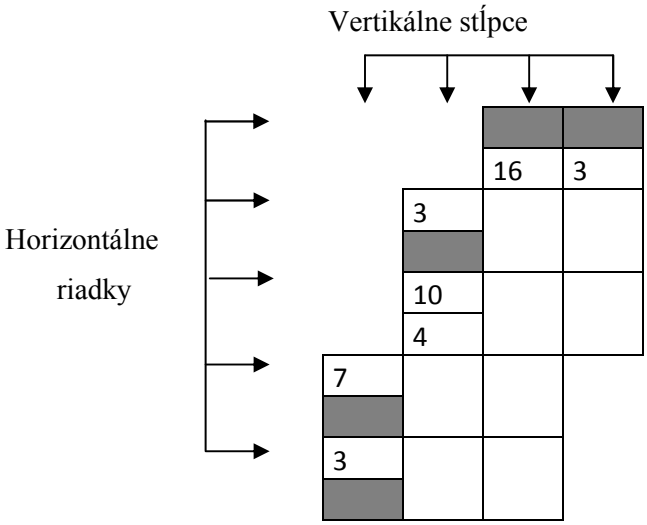
Pri testovaní jednotlivých aplikácií sa ukázalo, že aj malé detaily v implementácii pomocníka alebo zmeny pozadia môžu pokaziť alebo vylepšiť celkový dojem z hry. Na tento fakt bude potrebné myslieť pri vytváraní užívateľského rozhrania výslednej aplikácie (na základe štúdii z [8]).

4 Princíp hry kakuro

Ďalšou obdobou číselných logických hier je hra kakuro. Opis hry a princíp riešenia si ukážeme podrobnejšie v tejto kapitole.

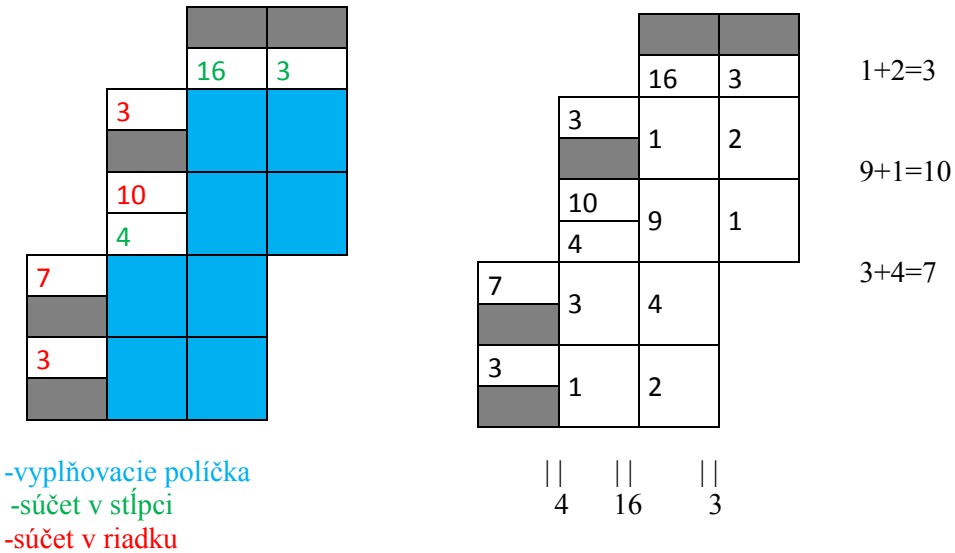
4.1 Pravidlá hry kakuro

Podobne ako sudoku aj kakuro môže mať rozličné veľkosti. Do štvorcov hlavolamu sa tak ako v sudoku dopĺňajú čísla od 1 po 9, navyše však musia dávať určitý súčet. Tak ako sudoku ani pravidlá hry Kakuro nie sú zložité. Hracie pole (Obr. 4. 1) je zložené zo štvorcových políčok, ktoré sú uložené v riadkoch (horizontálne) a v stĺpcoch (vertikálne). [1]



Obr. 4. 1.: Hracia plocha Kakura

Nad každým stĺpcom a naľavo každého riadka sa nachádza informačný štvorec, ktorý je rozdelený na dve časti hornú a dolnú. Ostatné políčka sa nazývajú vyplňovacie. Horná časť informačného štvorca nás informuje aký súčet dostaneme(ak správne dosadíme hodnoty do vyplňovacích políčok) **v riadku**. Dolná časť nás informuje aký súčet dostaneme **v stĺpci**. Ďalším pravidlom, ktoré platí pre hru kakuro je, že vyplňovanie políčka smie byť vyplňované výhradne číslicami z množiny {1,2,3,4,5,6,7,8,9}.



Obr. 4.2.: Hracia plocha Kakura

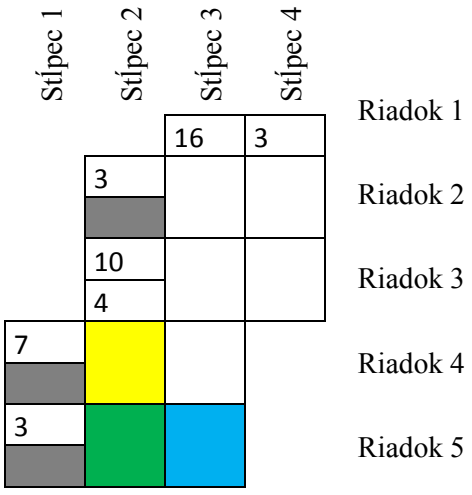
Ak si zhrnieme všetky podmienky zistime, že pre riešenie hry kakuro platia tieto 3 jednoduché pravidlá:

1. Vyplňovanie políčka vyplňujeme výhradne číslicami z množiny {1,2,3,4,5,6,7,8,9}.
2. Súčet vyplnených políčok v riadku sa musí rovnať hornej hodnote informačného políčka.
3. Súčet vyplnených políčok v stĺpci sa musí rovnať dolnej hodnote informačného políčka.
4. Každé číslo sa v jednom riadku a stĺpci môže nachádzať iba raz (na základe štúdii z [2] a [7]).

4.2 Riešenie hry kakuro

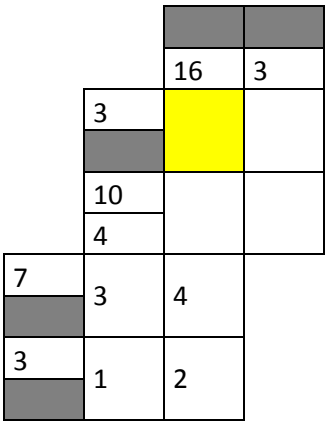
Oboznámili sme sa s pravidlami hry kakuro. No samotná znalosť pravidiel nezaručuje, že hru dokážeme aj úspešné vyriešiť. Preto si teraz ukážeme základné pravidlá ako postupovať pri riešení. Vyplníme krížovku nižšie. Hoci je krížovka jednoduchá, technika a postup riešenia sa uplatňuje aj pri riešení zložitých krížoviek. Pri riešení sa postupuje nasledovným spôsobom.

Začíname si všímať riadky alebo stĺpce, ktoré majú najmenší alebo najväčší počet políčok. Presnejšie hľadáme súčty, pre ktoré existuje najmenší počet kombinácií, takzvané jednoznačné súčty (pozri kapitolu 5.1). [2]



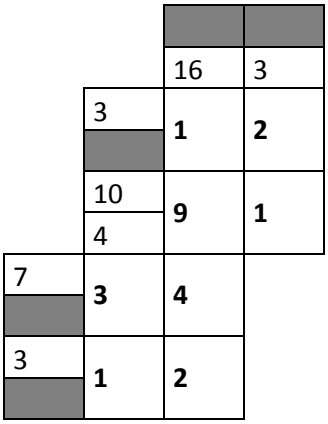
Obr. 4. 3.: Hracia plocha Kakura

Všimnime si druhý stĺpec a piaty riadok. Ak chceme dodržať pravidla kakuro, tj. v riadku a stĺpcoch sa čísla nemôžu opakovať a doplniť môžeme len čísla od 1 po 9. Z toho vyplýva, že štvorku a trojku na dvoch políčkach môžeme napísať jediným spôsobom (1+3=4, 2+1=3). Spoločným číslom oboch súčtov je jednotka, z toho vyplýva, že do zeleného políčka doplníme číslo jedna. Ďalej je jasné, že do žltého políčka príde 3, do modrého 2 a do políčka nad modrým príde číslo 4 (3+4)=7. [2] , [3]



Pozrime sa na druhý riadok, políčko vyznačené žltou farbou. Z informačného políčka naľavo vieme, že môžeme doplniť iba číslo 1 alebo 2 (1+2=3). Avšak číslo 2 sa už v stĺpci nachádza, preto ak chceme dodržať pravidla hry, môžeme doplniť jedine číslo 1. Ďalšie čísla doplníme veľmi jednoducho. Napravo od žltého políčka príde 2, pod žlté políčko príde 9, keďže 1 + 2 + 4+ +9 = 16 a vedľa deviatky príde číslo 1 (9+1=10). [2] ,[3]

Obr. 4. 4.: Hracia plocha Kakura



Úspešne sme vyplnili všetky políčka. Môžeme si všimnúť, že v žiadnom riadku ani stĺpci sa čísla neopakujú. Čísla sú z množiny {1,2,3,4,5,6,7,8,9} a súčty v riadkoch a stĺpcoch sa zhodujú so súčtami v informačných políčkach.
Z toho vyplýva, že sme našli správne a jediné riešenie hry kakuro. [2], [3]

Obr. 4. 5.: Hracia plocha Kakura

Ukázali sme si, že princíp riešenia hry kakuro nie je vôbec zložitý. Našou úlohou však bude tento postup zdokonaliť a algoritmizovať. Analýzou, návrhom algoritmu a samotným vypracovaním sa budem zaoberať v nasledujúcich kapitolách.

5 Návrh algoritmu

5.1 Tabuľka súčtov

Skôr ako sa pustime do samotnej dekompozície problému na pod problémy , vysvetlime si jednotlivé pojmy, ktoré budem pri vysvetľovaní problematiky používať.
Pojem: tabuľka súčtov, jednoznačné súčty, eliminačné súčty, číselné kombinácie súčtov, eliminované čísla a čísla, ktoré sa musia použiť.

Tabuľka súčtov (Tab. 5.1) - každý riadok tejto tabuľky sa začína počtom buniek, za ktorým nasleduje súčet, ktorý sa na bunkách nachádza (napríklad: 2s4 znamená že súčet 4, sa nachádza na 2 bunkách)
Tabuľku ďalej tvoria číselné kombinácie súčtov, eliminované čísla a čísla, ktoré sa musia použiť (napríklad: číselná kombinácia súčtu 4 na 2 bunkách je 13 eliminované čísla sú teda čísla: 2456789 a čísla, ktoré sa musia použiť sú čísla 13).

Jednoznačné súčty - sú súčty, pre ktoré existuje jediná číselná kombinácia súčtu (sú to súčty ako napr. 3 na 2 bunkách, pretože jediný možný spôsob ako dostať súčet 3 na dvoch bunkách, je sčítaním 1+2, alebo 4 na dvoch 2 bunkách 1+3)

Eliminačné súčty - sú súčty, ktoré nepatria medzi jednoznačné súčty a súčasne eliminujú aspoň jedno číslo (súčet 20 na 3 bunkách je eliminačný keďže eliminuje čísla 1 a 2)

Tab. 5.1.:Časť tabuľky súčtov dobre ilustruje jednotlivé pojmy.

Súčet	číselná kombinácia súčtu	eliminované	použité
2s3	12	3456789	12
2s4	13	2456789	13
2s5	14 23	56789	
.....			
2s16	79	1234568	79
2s17	89	1234567	89
3s6	123	456789	123
.....			
3s14	149 158 167 239 248 257 347 356		
3s15	159 168 249 258 267 348 357 456		
.....			
3s19	289 379 469 478 568	1	
3s20	389 479 569 578	12	
.....			

jednoznačný súčet eliminačný súčet

Celú tabuľku nájdeme v prílohe B.

5.2 Dekompozícia algoritmu

V predošlých kapitolách sme prebrali pravidla hry a spôsob riešenia hry kakuro. Kvôli úspešnému naprogramovaniu samotného algoritmu bude potrebné správne použiť dekompozíciu a celý algoritmus rozčleniť na menšie problémy. Jednotlivými problémami, s ktorými sa pri programovaní logickej hry kakuro stretneme sú:

- navrhnúť generovanie hracej plochy (generátor)
    - navrhnúť náhodné generovanie tvaru hracej plochy
    - navrhnúť náhodné vyplnenie hracieho poľa číslami
- navrhnúť algoritmus, ktorý vyrieši dané zadanie (riešiteľ, solver)
- navrhnúť spôsob generovania rôznych obtiažností
- navrhnúť užívateľské rozhranie hry (GUI, Graphic User Interface)

Generovanie kakura spadá do kategórie NP- úplných problémov. To znamená že v súčasnosti neexistuje algoritmus ktorý dokáže riešiť úlohu v polynomialnom čase. Časová zložitosť algoritmov NP problémov rastie neúmerným spôsobom ($t= n^x$) . Kým hraciu plochu o veľkosti 9x9 je možné vyriešiť do niekoľkých sekúnd, môže trvať niekoľko desiatok minút kým vyriešime plochu o veľkosti 15x15. Aj z tohto dôvodu je potrebne implementovať časovo čo najefektívnejšie metódy riešenia jednotlivých podproblémov. (na základe štúdii z [6])

Pri popise algoritmu budem často používať slovo riešiteľ (solver), nejde však o fyzickú osobu, ale o algoritmus, ktorý rieši vygenerovanú hraciu plochu.

5.3 Generovanie hracej plochy

Po vygenerovaní tvaru hracieho poľa nie je možné povedať či je toto pole použiteľné, preto je potrebné doplniť do poľa súčty a pokúsiť sa ho vyriešiť.

```
Kým nemáme jednoznačné riešenie {
        Náhodné_vygenerovanie_tvaru();                  // generator

        Kým nemáme jednoznačné riešenie alebo kým neusúdime, že pre daný tvar neexistuje
        riešenie{
                Náhodné_doplnenie_čísel();              // generator
                Pokus_o_jednoznačné_vyriešenie_kakura(); // solver
        }

}
```

Algoritmus 5.1.: Pseudokód algoritmu pre generovanie hracieho poľa kakuro

Obmedziť časovú náročnosť algoritmu môžeme nasledujúcimi spôsobmi:

1. Vhodná voľba generátora tvaru plochy.

   Ideálnym stavom je, aby pre každý tvar, ktorý z generátora tvaru dostaneme bolo možné nájsť jednoznačné riešenie kakura. Viac o generovaní tvaru v kapitole 5.3.1.

2. Vhodné doplňovanie náhodných čísel.

   Doplňovanie čísel by malo byť náhodné, no podľa určitých pravidiel tak ,aby bolo možné nájsť jednoznačné riešenie čo najskôr. Viac o náhodnom vyplňovaní v kapitole 5.3.2.

3. Implementácia čo najviac logického riešiteľa.

   Výsledné riešenie by sa malo hľadať čo najrýchlejšie a to dosiahneme implementovaním čo najlepšej logiky. Čím menej čísel musíme dosadiť hrubou silou, tým rýchlejší riešiteľ máme. Viac v kapitole 5.3.

5.3.1 Generovanie tvaru hracej plochy

V tejto časti si povieme niečo o generovaní hracích polí. Hracie polia je možné generovať buď to algoritmom, generovaním zo súboru (šablónami) alebo užívateľským vytváraním. Každý spôsob má svoje výhody aj nevýhody.

Spôsoby generovania hracieho poľa pre hru kakuro:

- generovanie algoritmom
- užívateľské vytváranie
- generovanie zo súboru (šablónami)

Pri generovaní hracieho poľa je potrebné z estetických ako aj praktických dôvodov dodržiavať nasledujúce pravidlá:

1. Minimálna dĺžka riadku, stĺpca je 2
2. Maximálna dĺžka riadku, stĺpca je 9.
3. Každý riadok a stĺpec musí začínať políčkom so súčtom.
4. V poliach by nemali vznikať uzavreté priestory.

5.3.1.1 Generovanie algoritmom

Najväčšou výhodou tohto spôsobu generovania je veľké množstvo rozličných tvarov hracích polí, ktoré je možné vygenerovať. Avšak vytvoriť algoritmus, ktorý by správne a zmysluplne generoval hracie pole nie je triviálny problém. Tento spôsob má síce len zopár, no zásadných nevýhod. Ako sme si už povedali generovanie tvaru hracieho poľa je NP úplný problém. Po vygenerovaní tvaru je potrebné nájsť pre tento tvar jednoznačné riešenie. Toto hľadanie zaberá určitý čas. Ani sebelepší algoritmus nemusí vygenerovať správny tvar na prvý pokus. Každým neúspešným generovaním sa doba, ktorú musí čakať užívateľ násobí. Nikdy si nemôžeme byť istí, na ktorý pokus sa vhodný tvar vygeneruje. Z tohto dôvodu nie je generovanie poľa algoritmom vhodné pre aplikácie, ktoré generujú hracie pole v reálnom čase. (na základe štúdií [6])

5.3.1.2 Generovanie zo súboru

Generovanie zo súboru využíva pre vytváranie hracieho poľa dopredu navrhnuté tvary (šablóny), ktoré vytvára programátor. Hlavnou výhodou tohto spôsobu je, že sa vyhneme tvarom, pre ktoré je nájdenie jednoznačného riešenia veľmi ťažké až nemožné. Týmto riešením výrazne obmedzíme časovú náročnosť výsledného programu. Tento spôsob je na implementáciu jednoduchší, nevznikajú tu problémy nelogického a nepriehľadného vytvárania poľa. Taktiež programátor môže vytvoriť tvary, ktoré sú užívateľsky zaujímavé. Nevýhodou tohto spôsobu je, že existuje obmedzený počet šablón, ktoré by sa pri dlhšom používaní programu mohli začať opakovať.
Riešením tohto problému by mohli byť aktualizácie, ktoré by pravidelne doplňovali nové zaujímavé šablóny.

5.3.1.3 Užívateľské vytváranie

Pri užívateľskom vytváraní vytvára hracie pole osoba, ktorá hru hrá. Užívateľ si sám zvolí veľkosť a tvar hracieho poľa. Vytváranie poľa nemusí byť pre väčšinu hráčov zaujímavé, preto pri implementácii tohto spôsobu by nemalo isť o jedinú možnosť vytvárania. Tento spôsob by mal byť len akýmsi rozšírením predchádzajúcich dvoch spôsobov.

Pre môj program som vybral generovanie zo súboru pomocou preddefinovaných šablón a užívateľské vytváranie.
Užívateľské vytváranie má v mojej implementácií dva spôsoby:

1. užívateľ vytvára len tvar hracej plochy
2. užívateľ vytvára tvar a napĺňa ho súčtami (užívateľ tak plní úlohu generátora)

5.3.2 Naplnenie hracieho poľa (generátor)

Po úspešnom vytvorení tvaru hracej plochy je potrebné hraciu plochu náhodne doplniť číslami.
Podobne ako pre generovanie tvaru aj pre doplňovanie čísel do poľa je problém NP úplný.
Existuje niekoľko spôsobov ako čísla do hracej plochy dopĺňať. Tieto spôsoby sa medzi sebou líšia obtiažnosťou implementácie a efektivitou. Spravidla platí, že čím je zložitejší spôsob na implementáciu, tým je algoritmus efektívnejší a rýchlejší.

Niekoľko spôsobov ako napĺňať hracie pole:

1. naplnenie poľa hrubou silou (Brute Force)
2. naplnenie poľa s využitím jednoznačných súčtov
3. naplnenie poľa s využitím jednoznačných a eliminačných súčtov

5.3.2.1 Naplnenie hracieho poľa hrubou silou

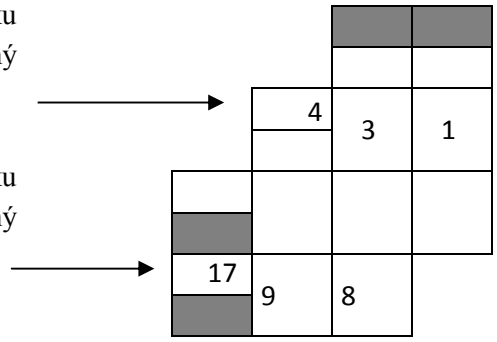
Spôsob, ktorý je najjednoduchší na implementáciu taktiež najjednoduchší na vypočetný výkon počítača, no všetko to je na úkor riešiteľa (solvera). Keďže platí, čím menej práce s generovaním hracej plochy, tým viac s riešením. Pri tomto spôsobe sa čísla do vygenerovaného poľa doplňujú náhodne bez akýchkoľvek pravidiel. Generovanie je veľmi rýchle, no šanca, že sa takouto metódou vygeneruje hracia plocha s jednoznačným riešením je veľmi malá. Kým riešiteľ(solver) nájde jednoznačné riešenie musia sa vygenerovať stovky až tisícky takýchto hracích plôch už pri relatívne malých rozmeroch (5x5). Z tohto dôvodu je nepredstaviteľné použiť takýto generátor pre rozmery 11x11 a viac.

5.3.2.2 Naplnenie hracieho poľa s využitím jednoznačných súčtov

Ďalším spôsobom ako do pripraveného tvaru dopĺňať čísla je spôsob s využitím jednoznačných súčtov. (Čo sú jednoznačné súčty sme sa detailne dozvedeli v kapitole 5.1.)
Ako takýto algoritmus pracuje:

Algoritmus zistí, že má doplniť súčet do riadku s počtom buniek 2.Nahodne vyberie jednoznačný súčet 4 (čísla 1 a 3) a pokúsi sa ho doplniť.

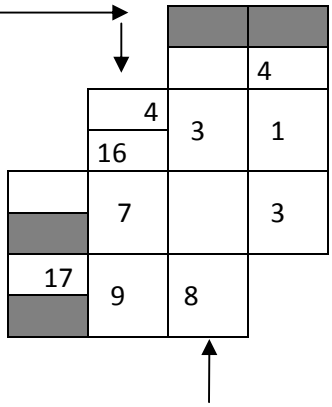
Algoritmus zistí, že má doplniť súčet do riadku s počtom buniek 2. Náhodne vyberie jednoznačný súčet 17 a pokúsi sa ho doplniť.



Obr. 5.1.: Hracia plocha Kakuro

Algoritmus zistí, že má doplniť súčet do riadku s počtom buniek 2, kde sa už nachádza číslo 3. Jediný možný a jednoznačný súčet, ktorý sem môže doplniť je 4 (čísla 1 a 3) .

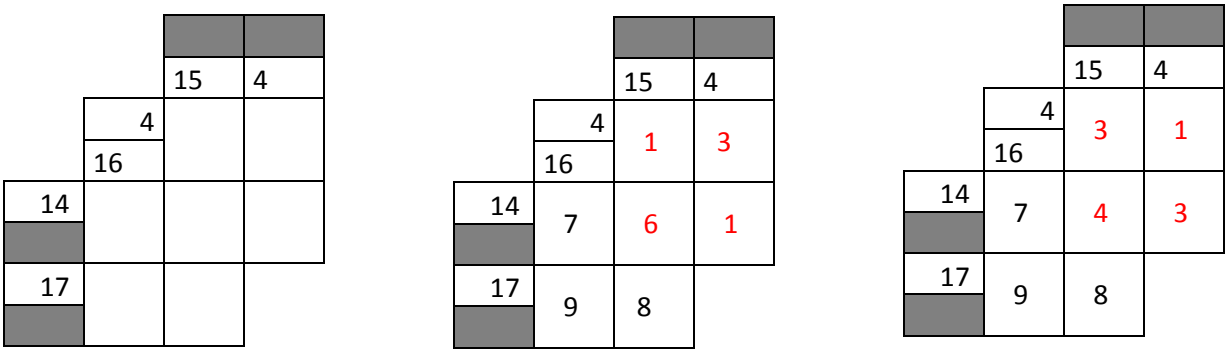
Algoritmus zistí, že má doplniť súčet do riadku s počtom buniek 2, kde sa už nachádza číslo 9. Náhodne vyberie jednoznačný súčet 16 a pokúsi sa ho doplniť.



Obr. 5.2.: Hracia plocha Kakura

Neexistuje žiadny jednoznačný súčet na 3 políčka, ktorý by obsahoval čísla 3 a 8 a algoritmus nemá žiadne ďalšie metódy, ktorými by zmysluplne dosadil ďalšie číslo, preto všetky nevyplnené políčka doplní náhodnými číslami. V našom prípade doplníme do políčka uprostred číslo 5 , čím vo vertikálnom smere dostaneme súčet 15 a horizontálnom smere dostaneme súčet 14.
Následne sa výsledná hracia plocha predá riešiteľovi (solveru).

Výsledná hracia plocha Riešenie hracej plochy



Obr. 5.3.: Hracia plocha Kakura

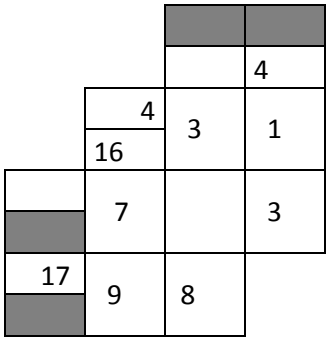
Riešiteľ (solver) zistí, že daná hracie plocha nemá jednoznačné riešenie, preto musí vygenerovanú hraciu plochu zahodiť a dať pokyn generátoru, aby sa pokúsil vygenerovať ďalšiu.

Oproti Brute force generátoru je tento algoritmus omnoho efektívnejší, keďže na vygenerovanie jednoznačného zadania pre veľkosť 5x5 mu priemerne stačí niekoľko desiatok pokusov.
Avšak pre veľkosti hracieho poľa 11x11 a viac sa počet pokusov stále pohybuje v tisícoch. Čo je pre program, ktorý potrebuje generovať hracie plochy real-time stále nepostačujúce. Riešenie ako toto generovanie zdokonaliť je v kapitole 5.3.2.3.

5.3.2.3 Naplnenie hracieho poľa s využitím jednoznačných a eliminačných súčtov

Doposiaľ najlepším známym spôsobom ako čo najefektívnejšie napĺňať hracie pole je využitie jednoznačných a eliminačných súčtov. Týmto algoritmom dokážeme vytvárať plochy o veľkosti 11x11 už po pár stovkách pokusov čo je veľký skok oproti akémukoľvek inému algoritmu. Práve z tohto dôvodu som vo svojom programe použil pravé tento algoritmus:

Algoritmus sa všade tam kde je to možné snaží dosadiť jednoznačné súčty. Až do bodu kým sa všetky možnosti pre umiestnenie jednoznačných súčtov nevyčerpajú, pracuje rovnako ako pracoval algoritmus s využitím jednoznačných súčtov.

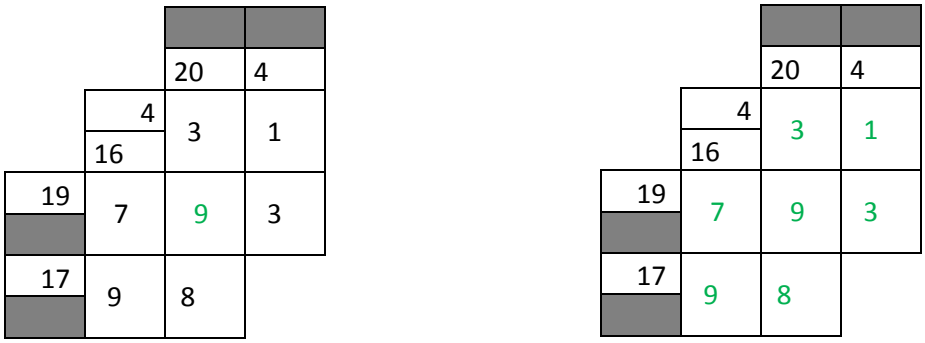


Obr. 5.4.: Hracia plocha Kakura

V tomto bode neexistuje žiadny jednoznačný súčet na 3 políčka, ktorý by obsahoval čísla 3 a 8 no na rozdiel od prechádzajúceho algoritmu, algoritmus s doplňovaním jednoznačných a eliminačných súčtov obsahuje metódu, ktorá skôr ako sa uchýli k doplneniu náhodného čísla skontroluje či neexistuje eliminačný súčet, ktorý by obsahoval čísla 3 a 8.

Algoritmus zistí, že má doplniť eliminačný súčet do riadku s počtom buniek 3, kde sa už nachádza číslo 3 a 8. Pozrie sa preto do tabuľky súčtov a pokúsi sa doplniť eliminačný súčet 20, číže číslo 9.

Vygenerovanú hraciu plochu doplnenú jednoznačnými a eliminačnými súčtami predáme solveru, ktorý zistí, že existuje jediné možné riešenie. V tejto chvíli je plocha s jednoznačným riešením pripravená na predanie do zobrazovacieho modulu, ktorý pole vykreslí a pripraví pre doplnenie užívateľovi.



Obr. 5.5.: Hracia plocha Kakura

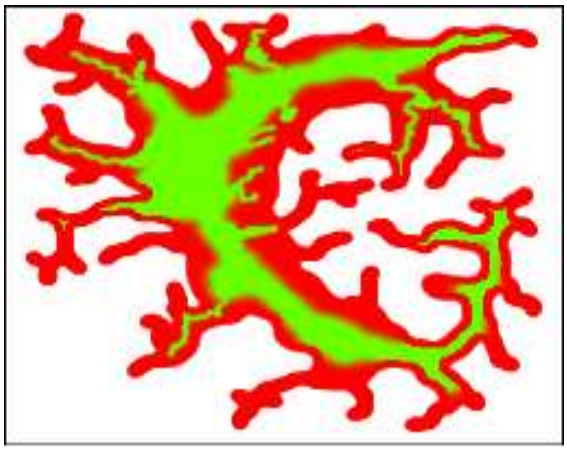
5.4 Riešiteľ hracieho poľa (solver)

Po úspešnom vygenerovaní hracej plochy a doplnení súčtami je potrebné overiť či je daná hracia plocha skutočne riešiteľná. Na rad nastupuje riešiteľ (ďalej už len solver). O základnom spôsobe riešenia hry kakuro sme hovorili v kapitole 4.2. V tejto kapitole si ukážeme ďalšie, zložitejšie postupy ako aj postup, ktorý je vo výslednom programe implementovaný.

Existujú dva spôsoby ako riešiť zadané hracie pole:

1. dosadzovaním čísel pomocou hrubej sily. (Brute force)
2. logickými postupmi

Pre lepšie pochopenie problematiky riešenia hry kakuro si všimnime obrázok Obr. 5.6. Obsah obdĺžnika ilustruje všetky možné zadania hry kakuro. (Obrázok je ilustračný keďže teoreticky počet hracích plôch kakuro je nekonečný). Bielou farbou sú vyznačené hry, pre ktoré neexistuje jednoznačné riešenie. Zelenou farbou sú vyznačené jednoznačné hry, ku ktorým je možné sa dostať logickými metódami. Červenou farbou sú vyznačené jednoznačné hry, no k ich riešeniu zatiaľ logické metódy neexistujú. Preto jediným spôsobom ako zistiť riešenie takýchto zadaní je použiť metódu „hrubej sily“ (na základe štúdii z [6]).

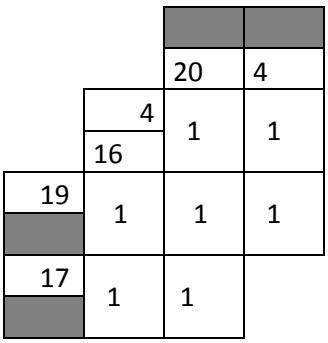


Obr. 5.6.: Zobrazenie riešení kakuro ([6])

Hlavnou výhodou algoritmu Brute force je, že dokáže nájsť každé platné zadanie hracej plochy kakura. Keďže doposiaľ neexistujú logické metódy k nájdeniu zadania, ktoré sú na obrázku znázornené červeňou farbou, ak chceme nájsť každé jedno platné (jednoznačné) zadanie musíme okrem logických metód použiť aj metódu brute force. Nevýhodou algoritmu brute force je jeho neefektivita. Bolo by maximálne neefektívne implementovať program, ktorý by hľadal riešenie len týmto spôsobom. Z toho vyplýva, že čím väčšie zastúpenie má v algoritme logika, tým je program efektívnejší.

5.4.1 Dosadzovanie čísel hrubou silou Brute force

Predstavme si, že máme metódu brute force, ktorá používa čísla od 1 po 9 bez akýchkoľvek logických postupov. Pokúsime sa vyriešiť toto zadanie:



Algoritmus si spočíta počet buniek, do ktorých musí dosadiť čísla. V našom prípade je to 7.
V prvom kroku vytvorí a dosadí vektor 1111111 až teraz sa dosadené čísla skontrolujú podľa známych pravidiel pre kakuro. Ak všetky pravidlá platia máme riešenie, no algoritmus tu nekončí, keďže musíme vyskúšať či je riešenie jednoznačné. Z tohto dôvodu by algoritmus skončil až po 7^9 (40 353 607) krokoch. Už pri malej veľkosti poľa je vidieť ako veľmi je tento spôsob neefektívny. V prípade, že by malo pole veľkosť 10x10 a obsahovalo by 70 políčok na vyplnenie algoritmus by skončil

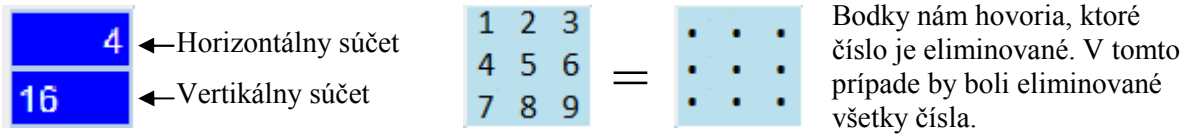
Obr. 5.7.: Hracia plocha Kakura

Ako je vidieť samotná metóda brute force sa na väčšie plochy kakura nedá použiť. Preto je nutné použiť metódu logiky, ktorú si popíšeme v nasledujúcej kapitole.

5.4.2 Dosadzovanie čísel logickými postupmi

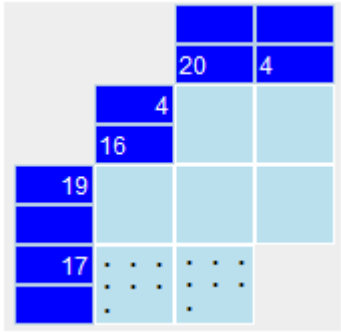
Jednoduchú metódu ako riešiť zadanie hry kakuro sme si už ukázali v kapitole 4.2. Tejto kapitole si ukážeme jednotlivé pokročilé univerzálne kroky, ktoré sú zaručenými metódami ako sa dopátrať jednoznačnému riešeniu hry. Všetky nasledujúce kroky sú algoritmizované a implementované v programe ktorý som vytvoril. Viac o implementácii nájdeme v kapitole 6.

Jednotlivé logické postupy budem nazývať eliminačné kroky. Keďže každý postup (eliminačný krok) eliminuje nejaké číslo, ktoré sa následne v bunke už nemôže nachádzať. Pri vysvetľovaní postupu budem používať nasledujúce zobrazenia:



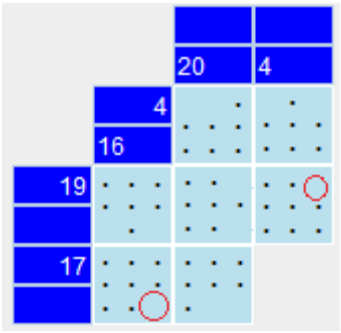
Bodky nám hovoria, ktoré číslo je eliminované. V tomto prípade by boli eliminované všetky čísla.

Obr. 5.8.: Zobrazenie používané pri vysvetľovaní



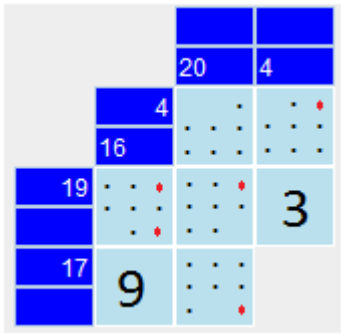
Obr. 5.9.: Hracia plocha

Pokúsime sa vyriešiť nasledujúce zadanie.
Krok 1:
Všimnime si súčet 17. Podľa tabuľky súčtov vieme, že ak máme súčet 17 na 2 políčkach automaticky môžeme vylúčiť čísla 1,2,3,4,5,6,7. Naznačíme to teda do krížovky.
Nasledujúci postup opakujeme pre každý jeden súčet.



Obr. 5.10.: Hracia plocha

Krok 2:
Po vylúčení všetkých čísel, ktoré sa v jednotlivých súčtoch nemôžu nachádzať vidíme, že v niektorých bunkách nám ostáva práve jedno prázdne miesto. V týchto prípadoch je jednoznačné, že jediné číslo, ktoré bude v týchto bunkách je číslo, nad ktorým sa prázdne miesto nachádza
(v stĺpci pod 4 to bude 3, v stĺpci pod 16 to bude 9).
Na tieto miesta teda doplníme čísla.



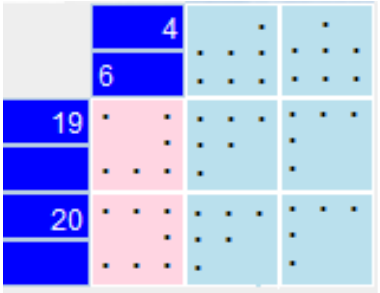
Obr. 5.11.: Hracia plocha

Krok 3:
Ak doplníme číslo do krížovky je potrebné zaistiť, aby sa v riadku ani stĺpci neopakovalo. Preto eliminujeme trojku a deviatku zo stĺpcov a riadkov, v ktorých sa už nachádza.

Krok 4:
Ak nám v riadku alebo stĺpci zostáva posledné nevyplnené číslo môžeme ho jednoducho dopočítať (v stĺpci pod štvorkou je to 4-3=1, v stĺpci pod 16 je to 16-9=8 ....).

Ak doplníme neeliminované a dopočítané čísla dostávame jednoznačné riešenie krížovky.
Na nájdenie riešenia by nám v tomto prípade stačilo týchto pár jednoduchých logických krokov.
No nie vždy si vystačíme iba s týmito preto si ukážeme zopár ďalších.

Predstavme si, že nastanú nasledujúce situácie:

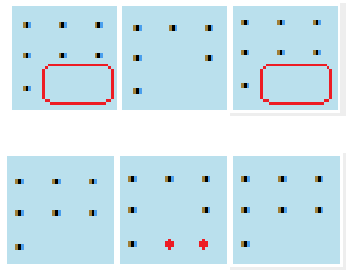


Obr. 5.12.: Výrez

Krok 5:
Všimnime si stĺpec pod súčtom 6. V tabuľke súčtov zistíme , že 6 sa dá zapísať nasledujúcimi číslami. Sú 4 možnosti ako naplniť bunky (od vrchu):

1. 1+5
2. 5+1
3. 2+4
4. 4+2

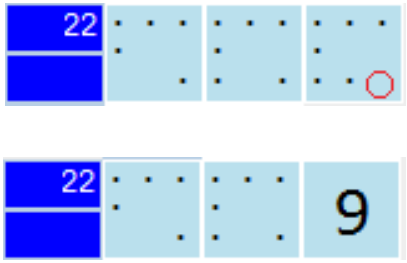
Keďže v prvej aj druhej bunke sme vylúčili číslo 1, prvá a druhá možnosť vypadáva. Takisto vypadáva štvrtá možnosť, keďže sme z druhej bunky vylúčili číslo 2. Z tohto dôvodu je jasné a jednoznačné, že v bunkách pod súčtom 6 budú čísla 2 a 4.



Krok 6:

V prípade takejto situácie si všímame, že v dvoch bunkách nám ostávajú dve rovnaké čísla, ktoré môžeme doplniť.(Do prvej a tretej bunky môžeme doplniť čísla 8 a 9). Ak tieto dve čísla doplníme do týchto buniek nebude môcť doplniť dané čísla nikde inde.
Z tohto dôvodu môžeme tieto čísla eliminovať zo všetkých ostatných buniek. (V našom prípade z prostrednej bunky eliminujeme čísla 8 a 9, čím jediné možné číslo, ktoré sme neeliminovali je číslo 5.)

Obr. 5.13.: Výrezy

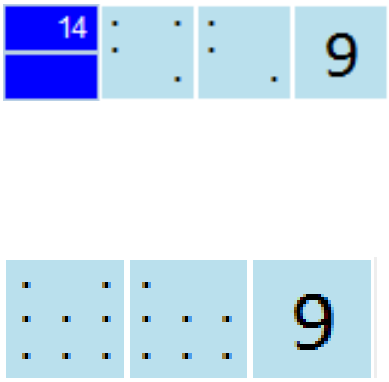


Krok 7:

V tabuľke súčtov si všimnime posledný stĺpec, ktorý nám hovorí aké čísla musia byť vo výslednom súčte použité. O súčte 22 môžeme s určitosťou povedať, že bude použité číslo 9.

Ak si všimneme 1. a 2. bunku vidíme, že číslo 9 už bolo eliminované. Preto posledná možnosť kde sa číslo 9 môže nachádzať je bunka 3.

Obr. 5.14.: Výrezy



Krok 8:

Máme ďalšiu zdanlivo neriešiteľnú situáciu. V tomto prípade opäť využijeme tabuľku súčtov. Všimnime si stĺpec s kombinačným vyjadrením súčtov. Vidíme, že číslo 14 sa dá zapísať s využitím nasledujúcich číselných kombinácií: 1+4+9, 1+5+8, 1+6+7, 2+3+9, 2+4+8, 2+5+7, 3+4+7, 3+5+6 .

Aj napriek relatívne veľkému množstvu možností je treba si uvedomiť, že v súčte 14 sa musí nachádzať číslo 9 (keďže je už doplnené). Z tohto dôvodu nám zostanú 2 varianty: 1+4+9 , 2+3+9(čísla 5,6,7,8,9 môžeme z buniek eliminovať).

Obr. 5.15.: Výrezy

Výhodou takto implementovaného postupu je časová zložitosť, keďže program v pár krokoch otestuje a eliminuje nevyhovujúce čísla. Ďalšou výhodou je odhad zložitosti krížovky. Čím viac logických krokov musíme pri riešení krížovky použiť, tým je náročnejšie zadanie. Nedokonalosťou tohto spôsobu je, že v súčasnej dobe neexistujú logické postupy na riešenie všetkých zadaní kakuro. Ako už bolo povedané, ak chceme nájsť všetky jednoznačné zadania hry, je potrebné implementovať hľadanie riešenia metódou hrubej sily. Teda logickými metódami eliminujeme čísla kým je to možné. Ak však dôjdeme k bodu kedy už nie je možné čísla eliminovať logikou, máme dve možnosti. Jednou z nich je dané zadanie zahodiť a pokúsiť sa vygenerovať nové, druhou možnosťou je doplniť číslo náhodné a ďalej pokračovať logickými metódami. Avšak, ako náhle číslo náhodne doplníme, nie je zaručená jednoznačnosť. Preto je potrebné, po úspešnom nájdení riešenia vrátiť sa ku kroku kde sme doplnili číslo náhodné a pokúsiť sa doplniť náhodne všetky ďalšie možnosti.

5.5 Obtiažnosť

Určovanie obtiažnosti v hre kakuro nie je triviálny problém. Je potrebné preriešiť mnoho krížoviek a dôkladne sa nad touto problematikou zamyslieť, keďže práve určenie obtiažnosti určuje celkový dojem z hry. Je nežiaduce, aby hra na zložitej obtiažnosti bola veľmi ľahká a ešte viac nežiaduce, aby hra jednoduchej obtiažnosti bola veľmi ťažká. Takáto zdanlivo malá chyba by mohla pokaziť celkový dojem z hry ako aj znechutenie z toho, že nie sme schopní vyriešiť jednoduchú krížovku o veľkosti 4x4. Môžeme si vybrať z nasledujúcich možností.

Možnosti ako ovplyvňovať obtiažnosti:

1. ovplyvňovanie obtiažnosti pri generovaní krížovky
2. ovplyvňovanie obtiažnosti pri riešení krížovky

5.5.1 Ovplyvňovanie obtiažnosti pri generovaní krížovky

Jedným zo spôsobov ako ovplyvniť obtiažnosť pri generovaní je použitie jednoznačných a eliminačných súčtov. Mohlo by sa totižto zdať, čím viac jednoznačných a eliminačných súčtov použijeme, tým jednoduchšia vygenerovaná krížovka bude. Preriešil a vygeneroval som mnoho krížoviek a z vlastných skúsenosti môžem povedať, že hoci krížovka obsahuje iba jednoznačné súčty, jej riešenie nemusí byt vôbec jednoduché. Jedine čo počet jednoznačných súčtov výrazne ovplyvňuje je efektívnosť pri nachádzaní jednoznačných riešení.
Uvediem príklad: Máme krížovku o veľkosti 4 x 4. Ak vkladáme jednoznačné a eliminačné súčty všade tam, kde je to možné, spravidla nájdeme krížovku s jednoznačným riešením do piatich pokusov. Ak za účelom zmeny obtiažnosti obmedzíme počet eliminačných a jednoznačných súčtov iba o 20%, je potrebné na nájdenie krížovky 2x až 3x viac pokusov. Je pravda, že čím menej jednoznačných a eliminačných súčtov krížovka obsahuje, tým väčšia pravdepodobnosť, že sa vygeneruje zložité riešenie, no nie je to pravidlom. Z dôvodu neistého výsledku a výrazného 100% až 200% zneefektívnenia algoritmu je obmedzovanie počtu jednoznačných a eliminačných súčtov v programe vylúčené.

Ďalším zo spôsobov ako ovplyvniť obtiažnosť pri generovaní je dĺžka doplňovacích polí a veľkosť hracej plochy. Majme jednoznačný súčet 4 na 2 políčkach. Musíme použiť čísla 1 a 3 existuje 2! možností (dva, 1+3 3+1) ako zapísať súčet 4 do dvoch buniek. Majme súčet 45 na 9 políčkach. Taktiež ide o jednoznačný súčet, keďže jediná možná kombinácia ako dostať súčet 45 je sčítať čísla 1,2,3,4,5,6,7,8,9. Avšak počet kombinácií ako rozmiestniť jednotlivé čísla na deviatich bunkách je v tomto príde mnohonásobne väčší 9!. Na rozdiel od metódy ovplyvňovania počtu jednoznačných a eliminačných súčtov je zaručené, že zadanie bude pôsobiť ťažšie ako opticky tak aj faktom, že pri doplňovaní súčtov bude potrebné pamätať si viac čísel a všímať si viac možnosti. Najdôležitejším faktom však je, že počet pokusov pri generovaní a hľadaní jednoznačného riešenia sa aj pri veľkosti 12 x 12 stále pohybuje v únosnej norme (priemerný čas na nájdenie riešenia je 5 sekúnd). Z tohto dôvodu je ovplyvňovanie zložitosti týmto spôsobom prijateľné a rozhodol som sa ho použiť aj v mojom programe.

5.5.2 Ovplyvňovanie obtiažnosti pri riešení krížovky

Ako sme si povedali v kapitole 5.5.1, obtiažnosť môžeme efektívne ovplyvniť už pri generovaní veľkosti poľa, ako aj počtom buniek, ktoré musíme doplniť. Avšak použitie iba tejto metódy by mohlo do hry priniesť určitú „nezábavnosť“, keďže menila by sa iba veľkosť hracej plochy a nie spôsob ako sa dopátrať k správnemu riešeniu. Riešením tohto problému je spojenie tejto metódy s metódou ovplyvňovanie obtiažnosti pri riešení krížovky. V predchádzajúcej kapitole 5.4.2 si môžeme všimnúť rôzne kroky logických postupov. Každý z nich má rôznu úroveň náročnosti. Kým kroky 1,2,3,4 sú triviálne, je jednoduché si ich logicky odvodiť, priamo vyplývajú z pravidiel, kroky 5,6,7,8 patria medzi náročnejšie. Preto napríklad, ak chceme vygenerovať jednoduchú krížovku, použijeme pri hľadaní riešenia iba kroky 1,2,3,4. Ak chceme krížovku ťažšiu, použijeme pri hľadaní riešenia vygenerovaného kakura okrem krokov 1,2,3,4 aj kroky 5 a 6 atď. Ak chceme najťažšiu, použijeme kroky 7,8, prípadne môžeme využiť metódu brute force.

Je potrebné si uvedomiť, že čím viac logických krokov máme v programe implementovaných, tým efektívnejšie a lepšie môžeme obtiažnosť ovplyvňovať. V prípade, že použijeme metódu brute force, patrí zadanie hry do kategórie veľmi ťažké, keďže jednotlivé čísla si nemôžeme eliminovať žiadnym spôsobom musíme ich naslepo tipovať a pokúšať sa dopátrať k riešeniu „šťastím“. Prvok „šťastia“ je však v logických hrách často nežiadúci, keďže s logikou nemá nič spoločné a na základe tohto som sa rozhodol tento spôsob čo najviac eliminovať tým, že generované krížovky sa dajú vyriešiť bez toho, aby sme si museli akékoľvek číslo tipovať.

5.6 Užívateľské rozhranie

Ďalšou neoddeliteľnou súčasťou každého počítačového softvéru, ktorého koncovým užívateľom je široká verejnosť je užívateľské rozhranie alebo GUI (Graphic User Interface). Dá sa povedať, že práve táto časť, hoci z pohľadu funkčnosti a efektívnosti algoritmu logických hier je nepodstatná, je pre väčšinu užívateľov najdôležitejším prvkom. Je dôležité, aby grafické prostredie pôsobilo elegantne, zábavne, intuitívne a aby bolo jednoduché sa v danom prostredí orientovať. Musíme si uvedomiť, že pri riešení väčších zadaní bude užívateľ nútený tráviť pred obrazovkou dlhšiu dobu. Sú dôležité prvky ako farby pozadia, veľkosť písma, ak chceme dosiahnuť, aby užívateľov pozadie po určitej dobe neprestalo baviť je potrebné vytvoriť niekoľko rôznych farebných štýlov.
Pri vytváraní grafického rozhrania by sme sa nikdy nemali spoliehať len na vlastný úsudok, preto je potrebné testovať program u viacerých užívateľov.

6 Implementácia algoritmu

Táto kapitola obsahuje popis spôsobu akým je implementovaný výsledný program. V kapitole 5 sme si celkový problém riešenia algoritmu rozložili na jednotlivé podproblémy. Teraz sa k týmto podproblémom vrátime a ukážeme si ako sme ich implementovali do výsledného programu.

6.1 Rozdelenie algoritmu do tried

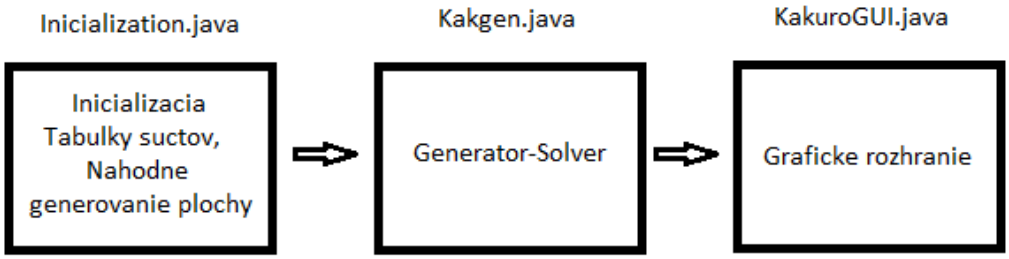
Vzhľadom k rozsiahlosti projektu bolo potrebné rozdeliť jednotlivé časti algoritmu do tried.
Hlavnou úlohou pri rozdeľovaní do tried bola lepšia prehľadnosť a zlepšenie orientácie v algoritme. V žiadnom prípade by toto rozdelenie nemalo spôsobiť zvýšenie časovej zložitosti algoritmu. Z týchto dôvodov som sa rozhodol rozdeliť algoritmus na nasledujúce triedy.

Inicialization.java
Je trieda, ktorá inicializuje tabuľky súčtov a náhodne generuje a vyberá tvar hracej plochy kakuro. O tom akým spôsobom sa tak deje sa dočítame v kapitolách 6.2 a 6.3.

Kakgen.java
V tejto triede je implementovaný hlavný výkonný algoritmus pre napĺňanie a hľadanie riešenia hry kakuro. Viac sa o samotnej implementácii dočítame v kapitolách 6.3 a 6.4.

KakuroGUI.java
A nakoniec táto trieda má za úlohu vytvoriť užívateľské rozhranie (Graphic User Interface, GUI), ktorým užívateľ ovláda hru, generuje a vytvára nové krížovky.

Grafické zobrazenie práce algoritmu. Ako prvé sa v triede Inicialization.java inicializujú tabuľky súčtov, náhodne sa vygeneruje hracia plocha. V ďalšej triede *Kakgen.java* Generator-Solver nájde jednoznačné zadanie pre danú plochu. Trieda *KakuroGUI.java* plochu vykreslí, pripraví pre doplnenie užívateľom. Podrobné správanie algoritmu vysvetlím nižšie v ďalších kapitolách.



Obr. 6.1.: Triedy programu

6.2 Tabuľka súčtov

V kapitole 5 si môžeme všimnúť nevyhnutnosť tejto tabuľky, keďže je potrebná či už pri generovaní nových zadaní, pri hľadaní riešenia, a dokonca ju využijeme aj v pomocníkovi v Grafickom rozhraní. Z tohto dôvodu je potrebné dostať sa k jednotlivým položkám tabuľky čo najrýchlejšie.
Najrýchlejšia metóda implementácie je prevedenie jednotlivých položiek do tvaru hashovacej tabuľky. Máme dve možnosti ako to urobiť. Buď vytvoríme program, ktorý bude tabuľku inicializovať automaticky alebo ju inicializujeme prvok po prvku ručne. Rozhodol som sa pre druhý spôsob, a to hneď z niekoľkých dôvodov.

Pre každý stĺpec tabuľky by bolo nutné vytvoriť program s exponenciálnou časovou zložitosťou. Vytvorenie takýchto programov nie je triviálny problém vzhľadom k rôznorodosti jednotlivých parametrov, ktoré je potrebné brať do úvahy. Zdanlivou výhodou by mohlo byť šetrenie pamäti keby sme pomocou funkcií dynamicky inicializovali iba tie čísla, ktoré vkladá generátor do krížovky. Avšak týmto krokom by sme výrazne spomalili celý algoritmus, keďže pri veľkostiach 6x6 a viac je na nájdenie jednoznačných krížoviek desiatky pokusov. Čísla by sa tak opakovane hľadali, vymazávali a vkladali do tabuľky.
Keďže časová náročnosť je v tomto prípade mnohonásobne dôležitejšia ako pamäťová, rozhodol som sa pre statickú inicializáciu všetkých prvkov v tabuľke súčtov.

Celá tabuľka súčtov je rozdelená do niekoľkých samostatných hashovacich tabuliek.
HashMap Komb_table; - číselná kombinácia súčtu
HashMap Used_table; - eliminované čísla
HashMap Must_table; - použité čísla
Tvar (Kľuč a hodnota) týchto tabuliek vyzerá nasledovne:
(“*Sucet***s***Pocet_buniek“,“hodnota“)* Príklad: *Komb_table*(“13s2“,“49 58 67“)
Tieto tabuľky sa využívajú výhradne pri hľadaní riešenia vygenerovaného zadania.

HashMap Uniq_table; - nachádzajú sa tu iba číselné kombinácie jednoznačných súčtov
HashMap Elim_table; - nachádzajú sa tu iba kombinácie eliminačných súčtov
Tvar (Kľuč a hodnota) týchto tabuliek vyzerá nasledovne:
(“*Poradove_cislo***s***Pocet_buniek“,“hodnota“)* Príklad: *Uniq _table*(“2s2“,“13“)
Tieto tabuľky sa využívajú výhradne pri generovaní nového zadania.

Inicializované tabuľky sa nachádzajú v triede *Kakgen.java.*
Inicializáciu jednotlivých tabuliek nájdeme v triede *Inicialization.java* nasledujúcim spôsobom:

Komb_table inicializuje metóda *Komb_table()*
Used_table inicializuje metóda *Used_table()*
Must_table inicializuje metóda *Must_table()*
Uniq_table inicializuje metóda *Uniq_table()*
Elim_table inicializuje metóda *Elim_table()*

6.3 Generovanie hracej plochy

6.3.1 Generovanie tvaru hracej plochy

Z dôvodu čo najväčšieho obmedzenia časovej náročnosti algoritmu som sa rozhodol generovať hraciu plochu pomocou predom pripravených šablón.

6.3.1.1 Náhodné generovanie tvaru hracej plochy

Náhodnosť generovania hracej plochy je zaistená nasledovne:
Tvary jednotlivých šablón sú uložené v triede *Inicialization.java*. Užívateľ si v grafickom rozhraní vyberie z niekoľkých možných veľkosti hracej plochy. Na výber je plocha od veľkosti 4x4 po veľkosť 12x12. Pre každú veľkosť existujú tri rôzne šablóny uložené kvôli rýchlemu vyhľadávaniu a prístupu k hodnotám v hashovacej tabuľke. Tvar (Kľúč a hodnota) týchto tabuliek vyzerá nasledovne:
(“*Poradove_cislo***size***Velkost_tabulky“,“vektortabulky“)*

S použitím triedy Random() vygenerujeme číslo z množiny {0,1,2}. Toto číslo je takzvané *Poradove_cislo* z reťazca, ktorý tvorí kľuč hesovacej tabuľky (“*Poradove_cislo***size***Velkost_tabulky“ , “vektortabulky“). Velkost_tabulky* je z triedy grafického rozhrania predaná do reťazca pre kľúč hashovacej tabuľky. Týmto spôsobom sme vytvorili náhodný kľúč, ktorým nájdeme k nemu priradený vektor pre vytvorenie tvaru hracej plochy.

6.3.1.2 Dátové uloženie hracej plochy

Pre reprezentáciu hracej plochy prichádzali do úvahy dva spôsoby. Prvá, logickejšia je možnosť reprezentovať plochu ako dvojrozmerné pole. Druhá možnosť je použiť jednoduché jednorozmerné pole. Kvôli jednoduchosti a menšej zložitosti implementácie som sa rozhodol pre reprezentáciu hracej plochy jednorozmerným poľom. Uloženú masku hracieho poľa nájdeme v triede *Inicialization()*. Reprezentuje ju vektor jedničiek a núl o veľkosti **size**x**size.** (Parameter **size** dostávame od užívateľa z grafického rozhrania)

Napríklad: Užívateľ zvolí veľkosť plochy 4x4 (size=4)
Spôsobom, ktorý je naznačený v kapitole 6.3.1.1 sa vyberie maska s kľúčom (2size4x4 ,0000001101110110). Na základe tohto vektoru sa vykreslí nasledovné pole:

Číslo 0 reprezentuje prázdne políčko, číslo 1 reprezentuje políčko kde sa bude nachádzať číslo na doplnenie. Všimnime si, že nie je potrebné naznačovať kde sa bude zobrazený súčet nachádzať, keďže je to všade tam kde sa začína nový stĺpec alebo riadok.

Obr. 6.2.: Hracia plocha

6.3.2 Náhodné naplnenie hracieho poľa.(Generátor)

Naplnenie hracieho poľa sa odohráva v triede Kakgen(). Ako som v kapitole 5.3.2 uviedol, rozhodol som sa použiť algoritmus s využitím jednoznačných a eliminačných súčtov, keďže tento algoritmus sa ukázal ako najefektívnejší a najlepší spôsob ako vygenerovaný tvar čo najúčinnejšie a najefektívnejšie napĺňať súčtami.

Ukážeme si ako samotné náhodné naplnenie súčtami prebieha.

V prvom kroku vytvoríme zoznam všetkých riadkov a stĺpcov, do ktorých budeme vkladať súčty. Zoznam uložíme do premennej typu *LinkedList*. Takto uložené riadky a stĺpce môžeme pohodlne náhodne premiešať pomocou metódy *Collections.shuffle* (*LinkedList*).

Program vyberie zo zoznamu prvý riadok alebo stĺpec. Podľa toho, koľko má riadok alebo stĺpec buniek, náhodne vyberie zo zoznamu (rovnako ako pri zozname riadkov a stĺpcov aj tu používam metódu *Collections.shuffle*(*LinkedList*)) jedinečných súčtov prvý súčet. Program najprv overí, či je daný súčet možné doplniť. Ak áno, doplní čísla súčtu do tabuľky, ak nie vyberie zo zoznamu ďalší jedinečný súčet a skúša vložiť čísla znovu. Po úspešnom alebo neúspešnom (ak sa nepodarí doplniť ani jeden jednoznačný súčet) doplnení súčtu program vyberie zo zoznamu riadku a stĺpcov ďalší prvok a opakuje postup, kým nie je zoznam prázdny. O prvotné doplnenie jednoznačných súčtov sa stará metóda *Kakurofill()*.

Následne sa krížovka doplnená jednoznačnými súčtami predá metóde *KakuroFillElim(),* ktorá sa rovnako ako metóda *Kakurofill()* pokúsi doplniť nie jednoznačné súčty, ale eliminačné. Ak aj toto doplnenie zlyhá, nevyplnené políčka sa doplnia náhodnými číslami.

V tejto chvíli máme pole naplnené číslami, toto pole sa predá ďalšej metóde *KakuroaddSum(),* ktorá doplní ku každému riadku a stĺpcu zodpovedajúci súčet a uloží ho do hashovacej tabuľky. Následne sa vygenerované a súčtami naplnené hracie pole predá riešiteľovi, ktorý sa pokúsi dané zadanie vyriešiť.

6.4 Riešiteľ hracieho poľa (Solver)

Pri riešení vygenerovanej hracej plochy algoritmus používa všetky logické metódy prezentované v kapitole 5.4.2. Metódu hľadania spôsobom brute force, keď všetky logické metódy zlyhajú, som sa rozhodol používať iba v obtiažnosti Hard, keďže takto vygenerované krížovky sa dajú riešiť len veľmi ťažko.

V užívateľskom rozhraní zadáme vygenerovať hracie pole, zavoláme teda metódu *KakGen_Solve(size)*, ktorá vygeneruje a vyrieši zadanie požadovanej veľkosti a obtiažnosti.
Pseudokód algoritmu hľadania riešenia pre vygenerované zadanie vyzerá nasledovne:

Generator-Solver zadania sa nachádza v triede *Kakgen.java* . Na vyriešenie krížovky sa používajú nasledujúce metódy, ich podrobný popis nájdeme v kapitole 5.4.2 a 5.4.1.

Inicializuj tvar hracieho poľa; // použijeme metódu *Grid_Mask()* z triedy *Inicialization*
Náhodne doplň čísla do šablóny; // použijeme metódu *Kakurofill(),KakuroFillElim()*

while (jednoznačné riešenie == false){

//nájdi riešenie algoritmu

KakElimination1(); //metóda, ktorá eliminuje čísla pomocou kroku1
KakTest1(); // metóda, ktorá eliminuje čísla pomocou kroku2
KakTest2(); // metóda, ktorá eliminuje čísla pomocou kroku3
KakElimination2(); // metóda, ktorá eliminuje čísla pomocou kroku4
If (obtiažnosť>easy) *KakElimination3();* // metóda, ktorá eliminuje čísla pomocou kroku5
If (obtiažnosť>easy) *KakElimination4();* // metóda, ktorá eliminuje čísla pomocou kroku6
If (obtiažnosť>medium) *KakElimination5();* // metóda, ktorá eliminuje čísla pomocou kroku7
If (obtiažnosť>medium) *KakElimination6();* // metóda, ktorá eliminuje čísla pomocou kroku8
If (obtiažnosť>medium) *KakTyprightnum();* // metóda brute force
If (máme jednoznačné riešenie) jednoznačné riešenie=true;
else Náhodne doplň čísla do šablóny;

Algoritmus 6.4.1.:Pseudokod generator-solvera algoritmu pre Kakuro

Okrem generovania krížovky môžeme v užívateľskom rozhraní krížovky aj vytvárať. Na výber sú 2 možnosti:

1. buď vytvoríme šablónu (tvar krížovky) spolu zo súčtami. // metóda *KakCheck_Gen_Solve()*
2. alebo vytvoríme iba tvar krížovky. // metóda *KakAdd_Gen_Solve()*

Metóda *KakCheck_Gen_Solve()* pracuje rovnako ako *KakGen_Solve(size),* akurát už nie je potrebné generovať masku náhodne, keďže masku tvaru hracej plochy vytvára užívateľ. Pri metóde *KakAdd_Gen_Solve()* nie je potrebné ani napĺňať plochu číslami, keďže prácu generátora robí užívateľ. Okrem požiadavky na jednoznačnosť musí tvar krížovky spĺňať pravidlá prezentované v kapitole 5.3.1.

Takto implementovaný algoritmus na hľadanie riešenia krížovky je veľmi efektívny. Rýchlo dokáže nájsť riešenie i v krížovkách, ktorých veľkosť plochy je väčšia ako 11x11. Avšak pri tajničkách o veľkosti 16x16 a viac sa stávalo, že užívateľ bol nútený čakať vyše 10 sekúnd. Keďže veľkosť tajničky 12x12 sa pre riešiteľov ukázala ako postačujúca, väčšie tajničky na výber v programe nenájdeme.

6.5 Obtiažnosť

V grafickom rozhraní si užívateľ môže vybrať z troch úrovní obtiažnosti: Easy, Medium, Hard.
Viditeľná zmena obtiažnosti je v rozmeroch hracieho poľa. Na obtiažnosti Easy je možné si vybrať z rozmerov hracieho poľa 4x4, 5x5, 6x6. Obtiažnosť medium ponúka rozmery plochy 7x7,8x8, 9x9. A nakoniec obtiažnosť hard ponúka plochy najväčšie 10x10,11x11, 12x12 .

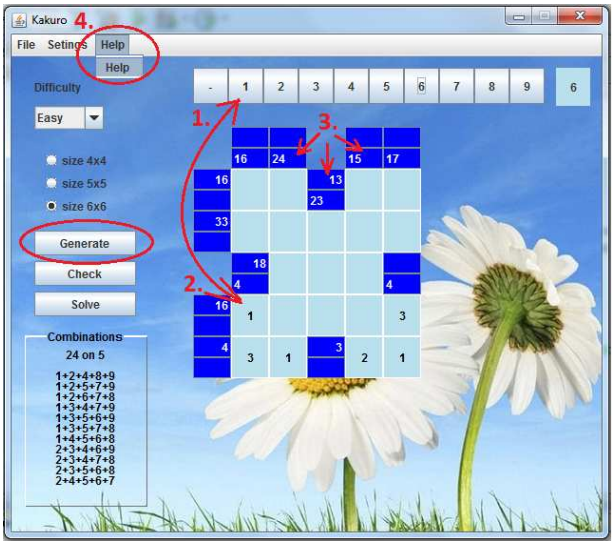
Okrem veľkosti hracej plochy sa mení aj maximálny počet buniek, do ktorých musíme súčty dopĺňať. V obtiažnosti easy dopĺňame súčty maximálne do piatich políčok. Šablóny hracieho poľa pre obtiažnosť medium sú stavané tak, aby do nich nebolo možné doplniť súčet dlhší ako 8 políčok. Obtiažnosť hard nemá žiadne obmedzenie na dĺžku súčtov, preto je možné doplniť ľubovoľný súčet.

Ďalšou metódou, ktorá ovplyvňuje obtiažnosť vygenerovanej hry je spôsob, ktorým hľadáme riešenie hry. Pri obtiažnosti easy sa na nájdenie jednoznačného riešenia hry používajú iba metódy *KakElimination1(), KakTest1() , KakTest2() a KakElimination2()*. To spôsobí, že ak sa aj vygeneruje zadanie s jednoznačným riešením na obtiažnosti easy, ktoré je ale veľmi náročné, algoritmus ho zahodí a pokúša sa vygenerovať zadanie nové, jednoduchšie. Pri obtiažnosti medium sa okrem týchto metód používajú metódy *KakElimination3(),KakElimination4()*. A nakoniec pri obtiažnosti hard sa používajú všetky dostupné metódy tj. *KakElimination5(),KakElimination6()* a spôsob brute force.

Dôvody prečo som sa rozhodol použiť pravé tieto spôsoby ovplyvňovania náročnosti ako aj podrobné vysvetlenie výhod a nevýhod jednotlivých spôsobov nájdeme v kapitole 5.5.

6.6 Užívateľské rozhranie

Vykresľovanie užívateľského rozhrania ako aj samotná funkčnosť jednotlivých prvkov je implementovaná v triede *KakuroGUI.java*. Pri implementácii som dbal na to, aby výsledné prostredie bolo čo najprívetivejšie pre užívateľov a bolo možné sa v ňom jednoducho a intuitívne pohybovať.
Po spustení programu sa spustí úvodné okno.
Na hornom okraji okna nájdeme menu, ktoré obsahuje niekoľko možností výberu. O jednotlivých možnostiach si povieme nižšie.
Na ľavej strane okna nájdeme ponuku s výberom obtiažnosti a veľkosťou hracej plochy. Po úspešnom nastavení týchto položiek je možné tlačítkom „Generate“ vygenerovať prvú hru.(Implicitne nastavenie je Easy , size 4x4). Zobrazí sa nám pole, do ktorého dopĺňame čísla nasledujúcim spôsobom:



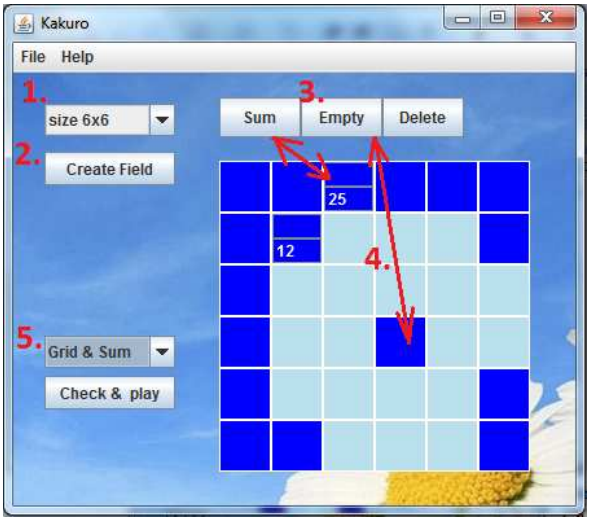
Obr. 6.3.: Printscreen Hry

1. klikneme na horné tlačítko s číslom, ktoré chceme do bunky dosadiť

2. následne nato klikneme do bunky kde chceme číslo dosadiť

Pri riešení hry sa môžeme oprieť o nástroje pomocníka, ako sú tlačidla na ľavej strane Check (skontroluje správnosť doplnených čísel), Solve (vyrieši zadanie), Tlačidlá súčtu (3. zobrazujú číselné kombinácie súčtov), alebo pomocníka (4. nájdeme v hornom menu pod ponukou „Help“).

Okrem generovania náhodnej plochy rôznych veľkosti nechýba možnosť generovania vlastnej plochy. Po kliknutí na menu položku „File“ a výberu možností „Make own grid“ (v pravom hornom rohu úvodného okna) sa zobrazí editor.



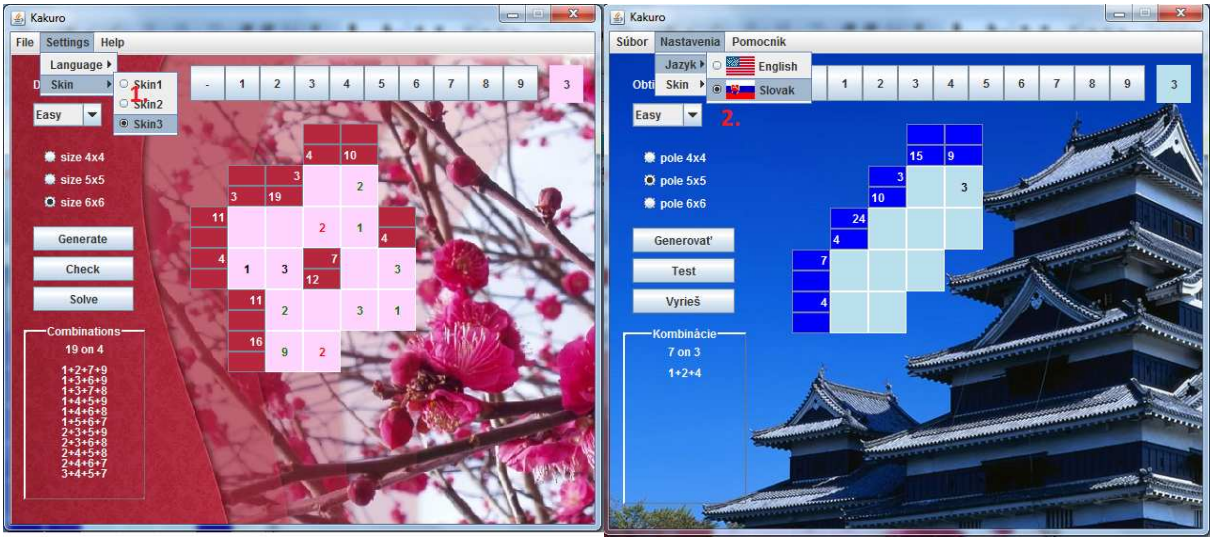
Obr. 6.4.:Printscreen Hry

Z možnosti veľkostí (1.) si vyberieme veľkosť hracej plochy a tlačidlom Create Field (2.) vytvoríme plochu. Hornými tlačidlami vložíme do poľa prázdne políčka a súčty (3.-4.).

Vyberieme si z možnosti kontroly(5). Ak chceme vytvoriť iba tvar hracej plochy vyberieme možnosť „Grid Only“. Ak chceme vytvoriť tvar so súčtami vyberieme možnosť „Grid & Sum“. Tlačidlom “Check & play“ skontrolujeme a vytvoríme plochu.

Prvým prvkom, ktorý zaisťuje príjemný a elegantný dojem z hry je pozadie. Užívateľ má na výber možnosť ľubovoľne meniť pozadie v menu, položke Settings. Na výber sú 3 farebné štýly –Skiny (1.).

Ako si môžeme všimnúť kvôli širšiemu spektru užívateľov, ktorým je hra určená som sa rozhodol rozšíriť menu o položku Language (2. čo znamená jazyk). V programe si teda okrem implicitne nastaveného anglického jazyka môžeme zvoliť aj jazyk slovenský.



Obr. 6.5.: Printscreen Hry

Pri implementácii užívateľského rozhrania som sa kvôli čo najlepšiemu prevedeniu nechával inšpirovať rôznymi užívateľmi. Veľké množstvo času bolo venované vylaďovaniu detailov, kvôli čo najlepšiemu pocitu z hry.

7 Testy

Dôležitá súčasť pri vytváraní programu bola fáza testov. Sprevádzala implementáciu od začiatku návrhu algoritmu až po konečné vylaďovanie Užívateľského rozhrania.
Samotnú fázu testov si teda môžeme rozdeliť do troch skupín.

1. Testovanie počas návrhu algoritmu.
   - Keďže sa jedná o náročný projekt, pred začatím programovania bolo potrebne overiť metódy riešenia hry kakuro prakticky. Mnoho hodín bolo venovaných nachádzaniu a overovaniu správnosti logických krokov prezentovaných v kapitole 5.4.2.

2. Testovanie počas implementácie
3. Testovanie a vyladenie Užívateľského rozhrania

7.1 Testy počas implementácie

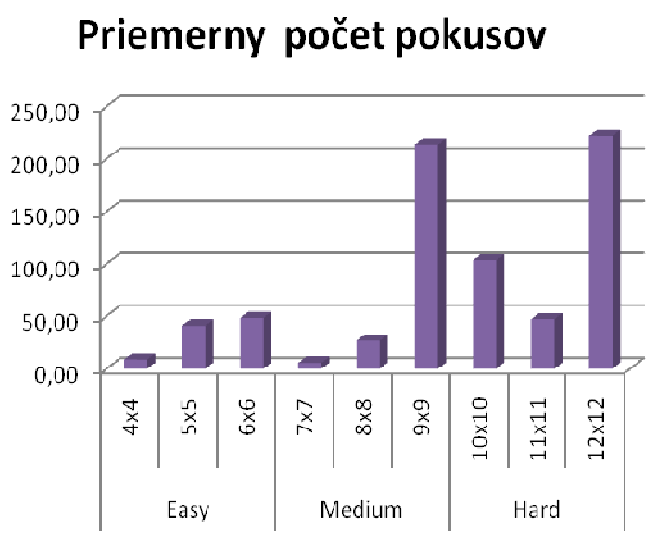
Náročnosť testovania bola v skutočnosti, že častokrát si nebolo možné overiť správnosť algoritmu postupne, ale až po naprogramovaní určitého celku. Ďalším problémom, s ktorým som sa musel počas testovania a vylaďovania algoritmu vysporiadať, bolo náročné hľadanie chýb. Keďže algoritmus musí pracovať náhodne, bolo mnohokrát ťažké zistiť, v ktorých konkrétnych prípadoch algoritmus padá , zacykľuje sa. Ďalšou podmienkou pre bezchybné fungovanie programu bolo vytvorenie šablón, ktoré by nespôsobovali zacykľovanie sa programu na dlhšiu dobu.

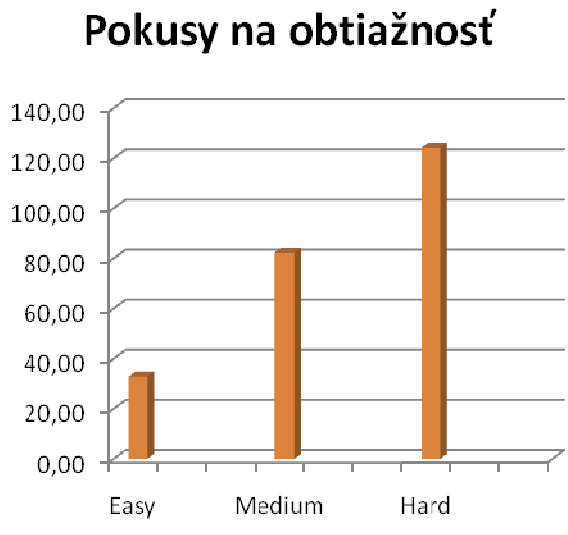
Tab.7.1.:Výsledky záverečných testov sú v nasledujúcej tabuľke:

Obtiažnosť	Easy			Medium			Hard		
Veľkosť plochy	4x4	5x5	6x6	7x7	8x8	9x9	10x10	11x11	12x12
Počet generovaní	44998	205306	242643	29661	134975	1070110	516720	235761	1109901
jednoznačné gen.	5000	5000	5000	5000	5000	5000	5000	5000	5000
Doba testovania	8 s	51 s	97 s	32 s	160 s	2050 s	1500 s	1020 s	6150 s
Priem. poč. pokusov	9,00	41,06	48,53	5,93	27,00	214,02	103,34	47,15	221,98
Priem. čas gener.	0,002s	0,010 s	0,019 s	0,006s	0,032 s	0,410 s	0,300 s	0,204 s	1,230 s
Priem. poč. na obtiaž.	32,86			82,32			124,16		
Priem. čas obtiažnosti	0,0104 sec			0,1495 sec			0,5780 sec		

V tabuľke a v grafe môžeme vidieť znázornený priemerný počet pokusov na vygenerovanie jednoznačného zadania.

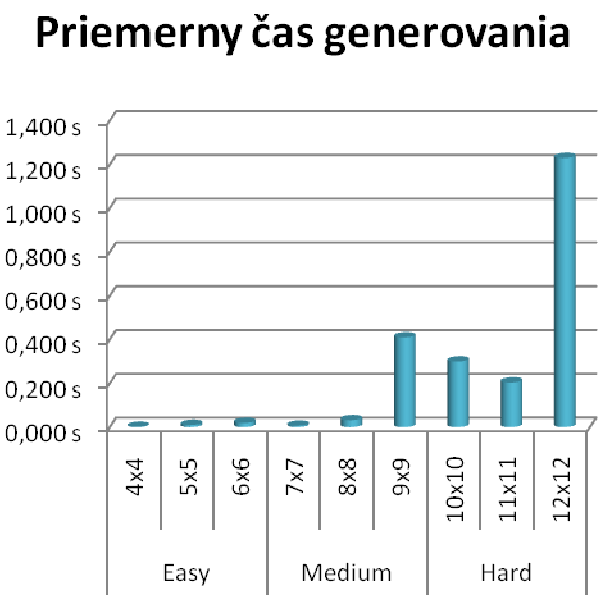
Graf 7.1.:Graficke zobrazenie počtu pokusov

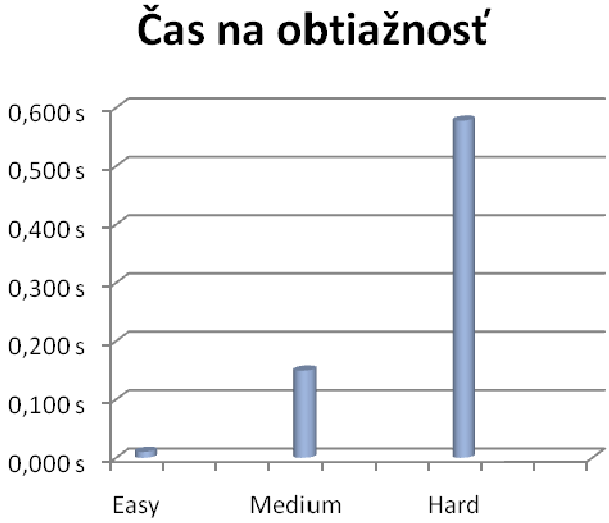




Všimnime si, že na vygenerovanie poľa o veľkosti 4x4 je v priemere potrebných 9 pokusov a na vygenerovanie poľa o veľkosti 7x7 je potrebných 5,9 pokusov. Rovnaký paradox si môžeme všimnúť u polí 6x6 a 7x7, 6x6 a 8x8, 9x9 a 10x10... S časti je to spôsobene šablónami hracieho poľa (pre niektoré šablóny sa jednoznačné riešenie hľadá ľahšie ako pre iné), no hlavným dôvodom sú metódy, ktorými sa riešenie hry hľadá. Kým v obtiažnosti easy sú to len 4 základne metódy, v obtiažnosti medium sa používa 6 metód a v obtiažnosti hard všetkých 8 logických metód prezentovaných v kapitole 5.4.2.

Graf 7.2.: Grafy zobrazujú čas potrebný na vygenerovanie jedného zadania.





Opäť si všimnime, že na vygenerovanie poľa o veľkosti 4x4 je v priemere potrebných 9 pokusov a na vygenerovanie poľa o veľkosti 7x7 je potrebných 5,9 pokusov. Napriek tomu priemerný čas generovania v prvom prípade je menši ako v druhom. Jednak je to spôsobené tým, že pri veľkosti poľa 4x4 je potrebné dopĺňať podstatne menšie množstvo buniek ako pri veľkosti poľa 7x7 a súčasne tým, že sa na nájdenie riešenia na obtiažnosti medium používa viac metód, teda viac strojového času než na obtiažnosti easy.

Algoritmus generator-solvera bol vystavený brute force testom ktoré mali odhaliť akúkoľvek možnosť zacyklovania sa. Ako všetky testy, aj toto testovanie prebehlo úspešne.

7.2 Testy užívateľského rozhrania

Pre čo najlepšiu objektívnu stránku tohto testovania bolo potrebné testovať aplikáciu u samotných užívateľov, pre ktorých je produkt určený. Prvým krokom tohto testovania bolo vytvorenie formulára, na základe ktorého by bolo jednoduché zistiť ktoré prvky v aplikácii je potrebné vylepšiť. K tomuto účelu bol vytvorený formulár „Anonymný dotazník k logickej hre Kakuro“.(Nájdeme ho v prílohe A.)
Ďalší postup je nasledovný:

1. vytvorenie verzie aplikácie
2. posúdenie užívateľov
3. na základe dotazníkov zdokonalenie aplikácie

Tento postup opakujeme kým nedosiahneme dostačujúcich výsledkov. V mojom prípade som tento postup musel opakovať celkom tri krát. Postupne boli vytvorené 3 verzie programu. Jednotlivé verzie programu a ich užívateľské testy si popíšeme v nasledujúcich kapitolách.

7.2.1 Verzia 1.0

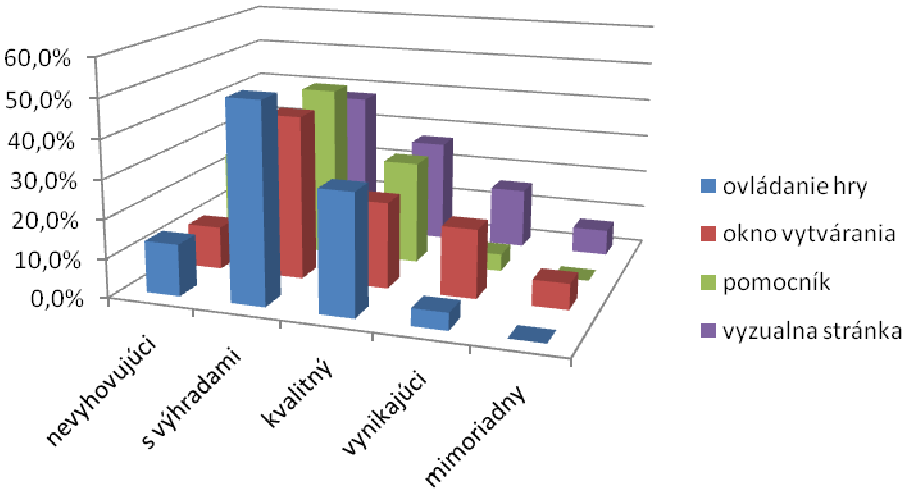
Aplikácia ma základne vlastnosti:

+generovanie hry rôznych obtiažnosti
+okno generovania vlastnej hry

Tab.7.2.: Tabuľka zobrazuje výsledky testov spokojnosti.
Vzorka 45 užívateľov.

	nevyhovujúci	s výhradami	kvalitný	vynikajúci	mimoriadny
ovládanie hry	13,3%	51,1%	31,1%	4,4%	0,0%
okno vytvárania	11,1%	42,2%	22,2%	17,8%	6,7%
pomocník	24,4%	44,4%	26,7%	4,4%	0,0%
vizuálna stránka	13,3%	37,8%	26,7%	15,6%	6,7%

Graf 7.3.: Výsledky testov spokojnosti.



7.2.2 Verzia 2.0

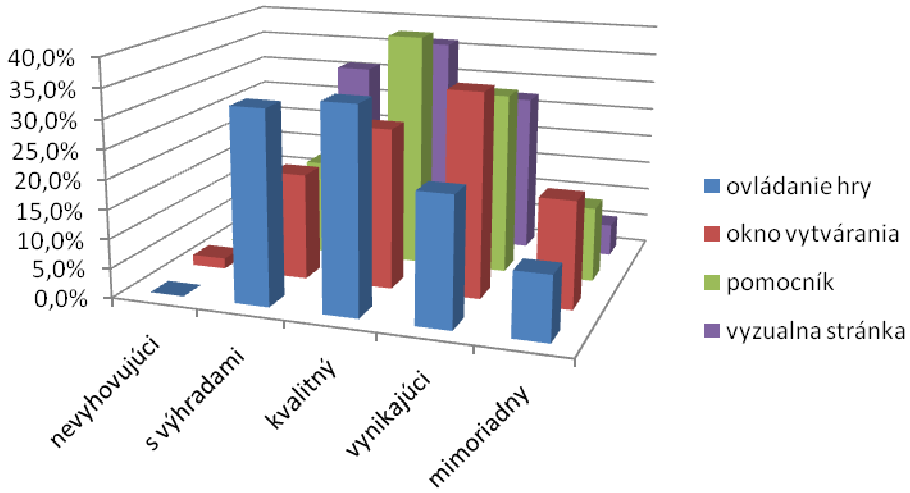
Oproti verzii 1.0 bola aplikácia vylepšená týmito vlastnosťami:

+úplne ovládanie programu len za pomoci myši
+tlačidlo kontroly riešenia
+tool tip text číselných kombinácii súčtov
+pomocník pri generovaní vlastnej hracej plochy
+farebný štýl hracieho poľa
+farebný štýl pozadia

Tab.7.3.: Tabuľka zobrazuje výsledky testov spokojnosti.
Vzorka 55 užívateľov.

	nevyhovujúci	s výhradami	kvalitný	vynikajúci	mimoriadny
ovládanie hry	0,0%	32,7%	34,5%	21,8%	10,9%
okno vytvárania	1,8%	18,2%	27,3%	34,5%	18,2%
pomocník	0,0%	16,4%	40,0%	30,9%	12,7%
vizuálna stránka	0,0%	30,9%	36,4%	27,3%	5,5%

Graf 7.4.: zobrazuje výsledky testov spokojnosti.



Na grafoch si môžeme všimnúť určité zlepšenie, avšak stále nie postačujúce. Z tohto dôvodu bolo potrebné vytvoriť ďalšie vylepšenia. Inšpiráciou bola okrem iného posledná dotazníková otázka.

7.2.3 Verzia 3.0

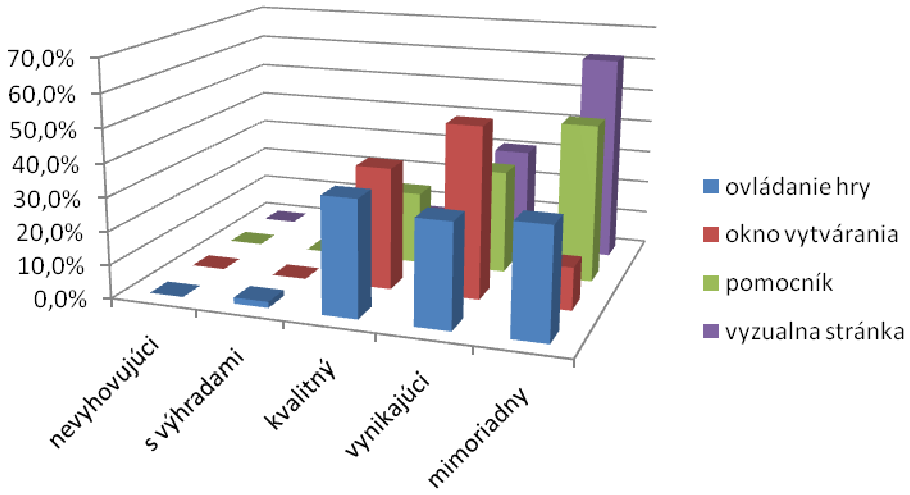
Súčasná podoba aplikácie. Úplný popis nájdeme v kapitole 6.6. Mnoho vylepšení oproti verzii 2.0 Dramatická zmena hlavne v grafickom štýle.

+test úspešného riešenia hry pri behu programu
+test duplikácie čísel v riadku, stĺpci
+víťazné okno (prevzaté z [10])
+zvuky tlačidiel (prevzaté z [14]))
+zvuk pri úspešnom vyriešení krížovky (prevzaté z [9])
+obrázok v pozadí (skin 1 a 2 prevzaté z [11] skin 3 prevzaté z [12])
+farebný štýl hracej plochy
+možnosť zmeny štýlu
+možnosť zmeny jazyka
+pri kliknutí na súčet zobrazenie možnosti čísel v súčte
+"loading okno" pri čakaní na novú krížovku
+pomocník s úplným návodom na ovládanie hry a hľadanie riešenia hry
+zdokonalenie chybových hlášok pri vytváraní hracej plochy
+možnosť vytvárania iba tvar hracej plochy
+farebné rozlíšenie chybných , správnych a doplnených čísel
+informačné okno zvoleného čísla pre doplnenie

Tab. 7.4.: zobrazuje výsledky testov spokojnosti.
Vzorka 55 užívateľov.

	nevyhovujúci	s výhradami	kvalitný	vynikajúci	mimoriadny
ovládanie hry	0,0%	1,8%	34,5%	30,9%	32,7%
okno vytvárania	0,0%	0,0%	36,4%	50,9%	12,7%
pomocník	0,0%	0,0%	21,8%	30,9%	47,3%
vizuálna stránka	0,0%	0,0%	7,3%	30,9%	61,8%

Graf 7.5.: zobrazuje výsledky testov spokojnosti.

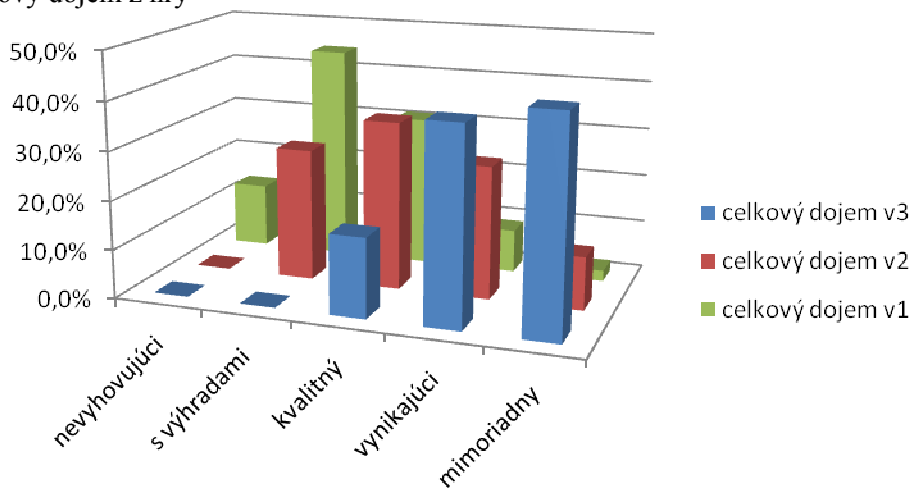


V tabuľke a na grafe si môžeme všimnúť výrazné zlepšenie vo všetkých aspektoch hry.

8 Záver

Prvým z cieľov tejto bakalárskej práce bolo zoznámenie sa s pravidlami logickej Hry Kakuro a spôsobmi riešenia. Hlavnou úlohou bolo navrhnutie generátor-solver tejto hry, ktorý by rýchlo a efektívne generoval zadania rôznych obtiažnosti. Ďalšou nemenej podstatnou úlohou tejto práce bolo vytvoriť užívateľský atraktívne a intuitívne grafické prostredie.
Pred samotnou implementáciou som problém dôkladne zanalyzoval, pri výbere spôsobov implementácii jednotlivých podproblémov som prihliadal na čo najväčšiu efektivitu a rýchlosť generovania. Výsledkom je aplikácia, ktorá rýchlo a efektívne generuje zadania požadovaných zložitostí. Keďže algoritmus generátor solvera funguje na logickej báze, nedochádza ku generovaniu náhodných zadaní, ktoré sú riešiteľné náhodnými, nekompetentnými, metódami. Na základe rozsiahlych testovaní môžeme vidieť, že priemerná doba generovania ľubovoľného zadania menšieho ako 12x12 neprekračuje 1,3 sekundy. Napriek tak výbornému priemernému času sa v niektorých prípadoch pri generovaní veľkosti 12x12 stáva, že jednoznačné riešenie je v dispozícii až po desiatich sekundách. Keďže u väčších veľkostí by tento časový interval mohol byť ešte väčší za pomyselný strop generovania veľkosti bola zvolená práve veľkosť 12x12.
Pre väčšie obtiažnosti by algoritmus mohol pôsobiť ťažkopádne (môžu nastať prípady kedy by bol užívateľ nútený čakať viac ako 10 sekúnd) riešením by bolo implementovať generovanie zadania do databázy v pozadí programu. Ďalšou možnosťou je zdokonalenie logiky algoritmu, prípadne zoptimalizovanie súčasnej logiky. Okrem testovania efektivity algoritmu, grafické rozhranie algoritmu prešlo rozsiahlym testovaním užívateľov. Program mal niekoľko verzií, ktoré mohli užívatelia testovať, podávať návrhy na zlepšenie. Väčšine návrhom som vyhovel, boli pridané funkcie ako zmena pozadia, zmena jazyka, víťazné okno a zvučka, rozšírené možnosti pomocníka a mnoho ďalších. Napriek odhodlaniu neboli implementované možnosť uloženia, nahratia hry, zlepšenie ovládania hry. Budúce prípadné zdokonalenie aplikácie vidím v implementácii práve týchto doplnkov ako aj zdokonalenie algoritmu hľadania jednoznačného zadania hry.

Graf 8.1.: Celkový dojem z hry



Z testov vyplýva, že vyše 83% testovaných užívateľov považuje výslednú aplikáciu za vynikajúcu až mimoriadnu. Hlavne vďaka príjemnému vzhľadu s dôrazom na detaily sa výsledná aplikácia rozšírila medzi užívateľmi, ktorí ju aj po skončení testovacej fázy radi zapnú a spríjemňujú si ňou chvíle odpočinku.

Literatúra

[1] MOORE Gareth, The Essential Book of Kakuro 2, Simon & Schuster, 2006 ISBN 0743299566, 9780743299565

[2] McMEEL Andrews, The original kakuro book, University of California Press, 2006, ISBN-13: 978-0-7407-6127-7

[3] MOORE Gareth, Kakuro – Nové japonské hlavolamy **,** FRAGMENT, 2005, ISBN 80-89210-85-6

[4] HODGES Wilfrid, Logic and Games, [online]. aktualizované 2009-01-08 [cit. 2010-4-9]. Dostupné na URL: <http://plato.stanford.edu/entries/logic-games/>

[5] Kolektív Autorov, Sudoku, [online].aktualizované 2009-11-24 [cit. 2010-04-9]. Dostupné na URL: <http://en.wikipedia.org/wiki/Sudoku>

[6] GROSSE Paul,Kakuro Trivia, [online], aktualizované 2010-1-6 [cit. 2010-04-20]. Dostupné na URL: <http://www.grosse.is-a-geek.com/kaktriv.html>

[7] Michael Mepham, Veľká kniha Japonských rébusů, Nakladatelství BB/Art, 2007, ISBN:978-80-7381-100-6

[8] kakuro.com, the home of Kakuro (cross sums) on the internet [online]. [cit. 2010-04-20]. Dostupné na URL: <http://www.kakuro.com/>

[9] Turner,Tina, Simply the best [online] , aktualizované 2009-6-3 [cit. 2010-04-25]. Dostupné na: < http://www.youtube.com/watch?v=ra12L1Bl0Z4&feature=channel>

[10] HacknMod, fireworks[online]. aktualizované 2009-7-23 [cit. 2010-04-25]. Dostupné na: < http://hacknmod.com/wp-content/uploads/2009/07/>

[11] Gerasimov Vlad, Spring in Japan II[online]. pridané 2005-5-7 [cit. 2010-04-29]. Dostupné na: < http://www.vladstudio.com/sk/wallpaper/?springinjapan2/>

[12] NatureWallpaper.eu, Nature Walpaper[online]. aktualizované 2009-11-21 [cit. 2010-04-29]. Dostupné na: < http://www.nature-wallpaper.info/>

[13] Kolektív Autorov,Magic Squares, [online].aktualizované 2009-11-24 [cit. 2010-03-11]. Dostupné na URL: < http://en.wikipedia.org/wiki/Magic_squares>

[14] Microsoft Corporation,Windows Seven Sounds , [software]. [cit. 2010-05-01]. Dostupné v adresári: < Windows\Media>

[15] Kolektív Autorov, Sudoku, [online].aktualizované 2009-11-24 [cit. 2010-03-11]. Dostupné na URL: < http://en.wikipedia.org/wiki/Nonogram>

Zoznam príloh

Príloha A. Testovací dotazník
Príloha B. Tabuľka súčtov
Príloha C. CD (zdrojové kódy, technická správa, spustiteľná aplikácia, programová dokumentácia, pomocník k hre)

Príloha A - Testovací dotazník

Anonymný Dotazník k Logickej hre kakuro.

Pri vyplňovaní formuláru uveďte prosím svoj objektívny názor bez ohľadu na Vašu rodinnú, alebo ľudskú príslušnosť ku mne. Za objektivitu veľmi pekne ďakujem.
Zaškrtnite jednu odpoveď. Napríklad takto: [X]

Ste spokojný s možnosťami ovládania hry?

- [ ] maximálne nespokojný
- [ ] viac nie
- [ ] môže byť
- [ ] viac áno
- [ ] maximálne spokojný

Ste spokojný s možnosťami okna vytvárania hry?

- [ ] maximálne nespokojný
- [ ] viac nie
- [ ] môže byť
- [ ] viac áno
- [ ] maximálne spokojný

Ste spokojný s možnosťami pomocníka v hre?

- [ ] maximálne nespokojný
- [ ] viac nie
- [ ] môže byť
- [ ] viac áno
- [ ] maximálne spokojný

Ste spokojný s možnosťami vizuálnou stránku hry?

- [ ] maximálne nespokojný
- [ ] viac nie
- [ ] môže byť
- [ ] viac áno
- [ ] maximálne spokojný

Celkový dojem z aplikácie.

- [ ] nevyhovujúci
- [ ] prijateľný s výhradami
- [ ] kvalitný
- [ ] vynikajúci
- [ ] mimoriadny

Ak by ste na algoritme mohli niečo zmeniť, vylepšiť. Čo by to bolo?

Príloha B - Tabuľka súčtov

Súčet	číselná kombinácia súčtu	elimin.	použité
2s3	12	3456789	12
2s4	13	2456789	13
2s5	14 23	56789	
2s6	15 24	36789	
2s7	16 25 24	789	
2s8	17 26 35	489	
2s9	18 27 36 45	9	
2s10	19 28 37 46	5	
2s11	29 38 47 56	1	
2s12	39 48 57	126	
2s13	49 58 67	123	
2s14	59 68	12347	
2s15	69 78	12345	
2s16	79	1234568	79
2s17	89	1234567	89
3s6	123	456789	123
3s7	124	356789	124
3s8	125 134	6789	1
3s9	126 135 234	789	
3s10	127 136 145 235	89	
3s11	128 137 146 236 245	9	
3s12	129 138 147 156 237 246 345		
3s13	139 148 157 238 247 256 346		
3s14	149 158 167 239 248 257 347 356		
3s15	159 168 249 258 267 348 357 456		
3s16	169 178 259 268 349 358 367 457		
3s17	179 269 278 359 368 458 467		
3s18	189 279 369 378 459 468 567		
3s19	289 379 469 478 568	1	
3s20	389 479 569 578	12	
3s21	489 579 678	123	
3s22	589 679	1234	9
3s23	689	123457	689
3s24	789	123456	789
4s10	1234	56789	1234
4s11	1235	46789	1235

Súčet	číselná kombinácia súčtu	elimin.	použité
4s11	1235	46789	1235
4s12	1236 1245	789	12
4s13	1237 1246 1345	89	1
4s14	1238 1247 1256 1346 2345	9	
4s15	1239 1248 1257 1347 1356 2346		
4s16	1249 1258 1267 1348 1357 1456 2347 2356		
4s17	1259 1268 1349 1358 1367 1457 2348 2357 2456		
4s18	1269 1278 1359 1368 1458 1467 2349 2358 2367 2457 3456		
4s19	1279 1369 1378 1459 1468 1567 2359 2368 2458 2467 3457		
4s20	1289 1379 1469 1478 1568 2369 2378 2459 2468 2567 3458 3467		
4s21	1389 1479 1569 1578 2379 2469 2478 2568 3459 3468 3567		
4s22	1489 1579 1678 2389 2479 2569 2578 3469 3478 3568 4567		
4s23	1589 1679 2489 2579 2678 3479 3569 3578 4568		
4s24	1689 2589 2679 3489 3579 3678 4569 4578		
4s25	1789 2689 3589 3679 4579 4678		
4s26	2789 3689 4589 4679 5678	1	
4s27	3789 4689 5679	12	9
4s28	4789 5689	123	89
4s29	5789	12346	5789
4s30	6789	12345	6789
5s15	12345	6789	12345
5s16	12346	5789	12346
5s17	12347 12356	89	123
5s18	12348 12357 12456	9	12
5s19	12349 12358 12367 12457 13456		1
5s20	12359 12368 12458 12467 13457 23456		
5s21	12369 12378 12459 12468 12567 13458 13467 23457		
5s22	12379 12469 12478 12568 13459 13468 13567 23458 23467		
5s23	12389 12479 12569 12578 13469 13478 13568 14567 23459 23468 23567		
5s24	12489 12579 12678 13479 13569 13578 14568 23469 23478 23568 24567		
5s25	12589 12679 13489 13579 13678 14569 14578 23479 23569 23578 24568		
5s26	12689 13589 13679 14579 14678 23489 23579 23678 24569 24578 34568		
5s27	12789 13689 14589 14679 15678 23589 23679 24579 24678 34569 34578		
5s28	13789 14689 15679 23689 24589 24679 25678 34579 34678		
5s29	14789 15689 23789 24689 25679 34589 34679 35678		
5s30	15789 24789 25689 34689 35679 45678		
5s31	16789 25789 34789 35689 45679		9
5s32	26789 35789 45689	1	89
5s33	36789 45789	12	789
5s34	46789	1235	46789
5s35	56789	1234	56789
6s21	123456	789	123456
6s22	123457	689	123457

Súčet	číselná kombinácia súčtu	elimin.	použité
6s23	123458 123467	9	1234
6s24	123459 123468 123567		123
6s25	123469 123478 123568 124567		12
6s26	123479 123569 123578 124568 134567		1
6s27	123489 123579 123678 124569 124578 134568 234567		
6s28	123589 123679 124579 124678 134569 134578 234568		
6s29	123689 124589 124679 125678 134579 134678 234569 234578		
6s30	123789 124689 125679 134589 134679 135678 234579 234678		
6s31	124789 125689 134689 135679 145678 234589 234679 235678		
6s32	125789 134789 135689 145679 234689 235679 245678		
6s33	126789 135789 145689 234789 235689 245679 345678		
6s34	136789 145789 235789 245689 345679		9
6s35	146789 236789 245789 345689		89
6s36	156789 246789 345789		789
6s37	256789 346789	1	6789
6s38	356789	124	356789
6s39	456789	123	456789
7s28	1234567	89	1234567
7s29	1234568	79	1234568
7s30	1234569 1234578		12345
7s31	1234579 1234678		12347
7s32	1234589 1234679 1235678		123
7s33	1234689 1235679 1245678		126
7s34	1234789 1235689 1245679 1345678		1
7s35	1235789 1245689 1345679 2345678		5
7s36	1236789 1245789 1345689 2345679		9
7s37	1246789 1345789 2345689		489
7s38	1256789 1346789 2345789		789
7s39	1356789 2346789		36789
7s40	1456789 2356789		56789
7s41	2456789	13	2456789
7s42	3456789	12	3456789
8s36	12345678	9	12345678
8s37	12345679	8	12345679
8s38	12345689	7	12345689
8s39	12345789	6	12345789
8s40	12346789	5	12346789
8s41	12356789	4	12356789
8s42	12456789	3	12456789
8s43	13456789	2	13456789
8s44	23456789	1	23456789
9s45	123456789		123456789